ΑΕΙ ΠΕΙΡΑΙΑ Τ.Τ.

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ Τ.Ε.

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Έξυπνο σπίτι με τη βοήθεια φωνητικών οδηγιών σε κινητή συσκευή IOS

Άγγελος Παπαγεωργίου & Ορέστης Ντιβιντάρι

Εισηγητής: Δρ Ιωάννης Έλληνας, Καθηγητής

ΑΘΗΝΑ
ΙΟΥΝΙΟΣ 2016

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο/Η κάτωθι υπογεγραμμένος/η ……,
του ……, με αριθμό μητρώου ……. φοιτητής/τρια του Τμήματος Μηχανικών Η/Υ Συστημάτων Τ.Ε. του Α.Ε.Ι. Πειραιά Τ.Τ. πριν αναλάβω την εκπόνηση της Πτυχιακής Εργασίας μου, δηλώνω ότι ενημερώθηκα για τα παρακάτω:

«Η Πτυχιακή Εργασία (Π.Ε.) αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο του συγγραφέα, όσο και του Ιδρύματος και θα πρέπει να έχει μοναδικό χαρακτήρα και πρωτότυπο περιεχόμενο.

Απαγορεύεται αυστηρά οποιοδήποτε κομμάτι κειμένου της να εμφανίζεται αυτούσιο ή μεταφρασμένο από κάποια άλλη δημοσιευμένη πηγή. Κάθε τέτοια πράξη αποτελεί προϊόν λογοκλοπής και εγείρει θέμα Ηθικής Τάξης για τα πνευματικά δικαιώματα του άλλου συγγραφέα. Αποκλειστικός υπεύθυνος είναι ο συγγραφέας της Π.Ε., ο οποίος φέρει και την ευθύνη των συνεπειών, ποινικών και άλλων, αυτής της πράξης.

Πέραν των όποιων ποινικών ευθυνών του συγγραφέα σε περίπτωση που το Ίδρυμα του έχει απονείμει Πτυχίο, αυτό ανακαλείται με απόφαση της Συνέλευσης του Τμήματος. Η Συνέλευση του Τμήματος με νέα απόφασης της, μετά από αίτηση του ενδιαφερόμενου, του αναθέτει εκ νέου την εκπόνηση της Π.Ε. με άλλο θέμα και διαφορετικό επιβλέποντα καθηγητή. Η εκπόνηση της εν λόγω Π.Ε. πρέπει να ολοκληρωθεί εντός τουλάχιστον ενός ημερολογιακού 6μήνου από την ημερομηνία ανάθεσης της. Κατά τα λοιπά εφαρμόζονται τα προβλεπόμενα στο άρθρο 18, παρ. 5 του ισχύοντος Εσωτερικού Κανονισμού.»

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα πτυχιακή εργασία ολοκληρώθηκε μετά από επίμονες προσπάθειες, σε ένα ενδιαφέρον γνωστικό αντικείμενο, όπως αυτό της επεξεργασίας κειμένου. Την προσπάθειά μου αυτή υποστήριξε ο επιβλέπων καθηγητής μου, τον οποίο θα ήθελα να ευχαριστήσω.

Ακόμα θα ήθελα να ευχαριστήσω τον κ. Μανώλη Λέανδρο για τις πολύτιμες συμβουλές του και την οικογένειά μου που θα ήθελε να τελειώσω τις σπουδές μου σε λιγότερο από οκτώ χρόνια.

Περίληψη

1. Το πρόβλημα των ακριβών HomeKit αξεσουάρ

Η Apple έχει πολύ αυστηρές διαδικασίες έγκρισης για όποιον κατασκευαστή θέλει να δημιουργήσει αξεσουάρ που κάνουν χρήση του πρωτοκόλλου Homekit. Ζητάει να δει προϋπολογισμούς, εγκαταστάσεις και άλλες τεχνικές λεπτομέρειες προτού δώσει πρόσβαση σε εταιρείες στην αρχιτεκτονική του πρωτοκόλλου Homekit, ενώ παράλληλα ζητάει και ένα υψηλό αντίτιμο περί τα 5000€ προκειμένου να τους το παράσχει. Κάτι το οποίο αυτόματα ανεβάζει την τιμή των αξεσουάρ από το στάδιο του R&D πριν καν μπει μια εταιρία στην δημιουργία αξεσουάρ! Αυτό κάνει απαγορευτική την δημιουργία αξεσουάρ homemade αξεσουάρ, που μπορεί να είχαν χαμηλότερη τιμή απόκτησης.

2. Η προτεινόμενη λύση σε αυτή την πτυχιακή

Στο πρόβλημα των ακριβών license για την κατασκευή homebrew Homekit αξεσουάρ την λύση έρχεται να φέρει το reverse engineer. Με την χρήση του προσομοιωτή Homekit αξεσουάρ που φτιάχτηκε από την Apple προκειμένου να βοηθήσει στην ανάπτυξη εφαρμογών χωρίς να είναι απαραίτητη η ύπαρξη κάποιου Homekit αξεσουάρ από τον developer της εφαρμογής και με ένα ip packet sniffer σε αυτή την εφαρμογή μπόρεσαν κάποιοι ταλαντούχοι ερευνητές να αποδομήσουν το πρωτόκολλο Homekit και να φτιάξουν έναν server χρησιμοποιώντας την γλώσσα NodeJS για να λειτουργεί και να παρουσιάζει accessories στο πρωτόκολλο Homekit που υποστηρίζουν οι συσκευές iOS. Αυτό σήμερα θα κάνει δυνατή την ανάπτυξη αξεσουάρ για αυτή την πτυχιακή τα οποία να έχουν ένα μέσο κόστος κατασκευής από €10-€20. Κόστος πολύ μικρότερο αυτού των οποιονδήποτε αξεσουάρ που εμπορεύονται εταιρείες ηλεκτρικών/ηλεκτρονικών. Αυτός ο server μαζί με τις τεχνολογίες του Arduino και της πλατφόρμας iOS θα δομήσουν το όλο έργο αυτής της πτυχιακής.

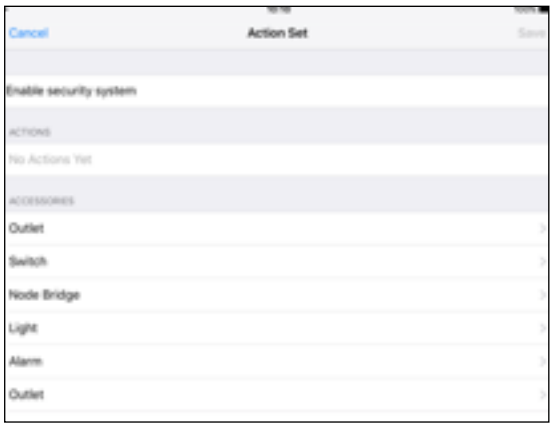
Περιεχόμενα

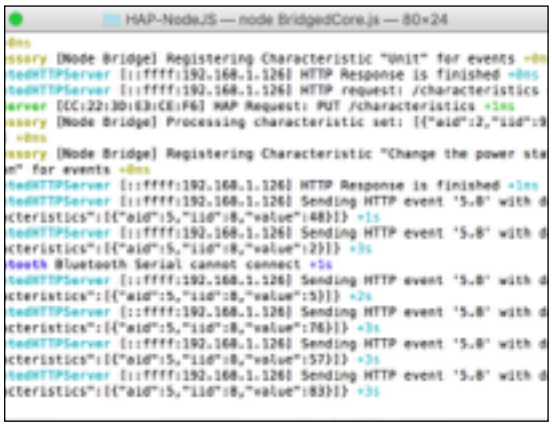
Κατάλογος Εικόνων

Κατάλογος Πινάκων

Κεφάλαιο 1

Εισαγωγή





Homekit Accessories

Η ιδέα πίσω από αυτό το project δημιουργήθηκε από το συνεχόμενο κύμα αξεσουάρ που χρησιμοποιούνται στο σπίτι μας για να διευκολύνουν την ζωή μας. Έτσι σκεφτήκαμε να χρησιμοποιήσουμε ότι είχαμε μάθει για τα ηλεκτρονικά μέσα στην διάρκεια των σπουδών μας προκειμένου να φτιάξουμε την δική μας σειρά αξεσουάρ για το σπίτι με τα λιγότερα δυνατόν χρήματα. Παρόμοια αξεσουάρ από εταιρείες κολοσσούς πωλούνται προς 50€-300€, είπαμε πως κάτι τόσο απλό στην κατασκευή δεν είναι δυνατόν να κοστίζει τόσο πολύ. Συνεπώς σκεφτήκαμε τον ευκολότερο και φθηνότερο τρόπο για να φτιάξουμε παρόμοια αξεσουάρ με αυτά των μεγάλων εταιρειών για τον 1/3 του κόστους. Χρησιμοποιώντας παράλληλα την τεχνολογία Homekit προκειμένου να κάνουμε αυτά τα αξεσουάρ ευκολότερα στον χειρισμό, ακόμα και μέσω φωνητικών εντολών από συσκευές iOS. Καταλήξαμε

λοιπόν στην δημιουργία τεσσάρων ελεγκτών οικιακών συσκευών. Έναν ελεγκτή διακοπτών, έναν ελεγκτή πολύπριζου, έναν ελεγκτή λάμπας και έναν ελεγκτή οικιακού συναγερμού. Επιλέξαμε τους συγκεκριμένους ελεγκτές καθώς θεωρήσαμε πως ελέγχουν πολύ συνηθισμένες οικιακές συσκευές, που υπάρχουν σε κάθε σπίτι, αλλά κυρίως γιατί ελέγχουν ενεργοβόρες συσκευές που πολλές φορές όταν φεύγουμε από το σπίτι ξεχνάμε να απενεργοποιήσουμε. Με τους ελεγκτές που δημιουργήσαμε εξαλείψαμε τον κίνδυνο υπερκατανάλωσης ρεύματος από αυτές τις συσκευές, μιας και πλέον είναι δυνατός ο απομακρυσμένος έλεγχος τους.

Κεφάλαιο 2

Κατασκευή

ΝΑ ΜΠΕΙ ΦΩΤΟ ΤΗΣ ΚΑΤΑΣΚΕΥΗΣ

Χρησιμοποιήσαμε ένα πρότυπο για όλα τα αξεσουάρ που κατασκευάσαμε προκειμένου να κάνουμε πιο εύκολη την υλοποίηση του project αλλά και να καλύψουμε πολλαπλές εφαρμογές αξεσουάρ.
Η κατασκευή αυτή αποτελείται από :

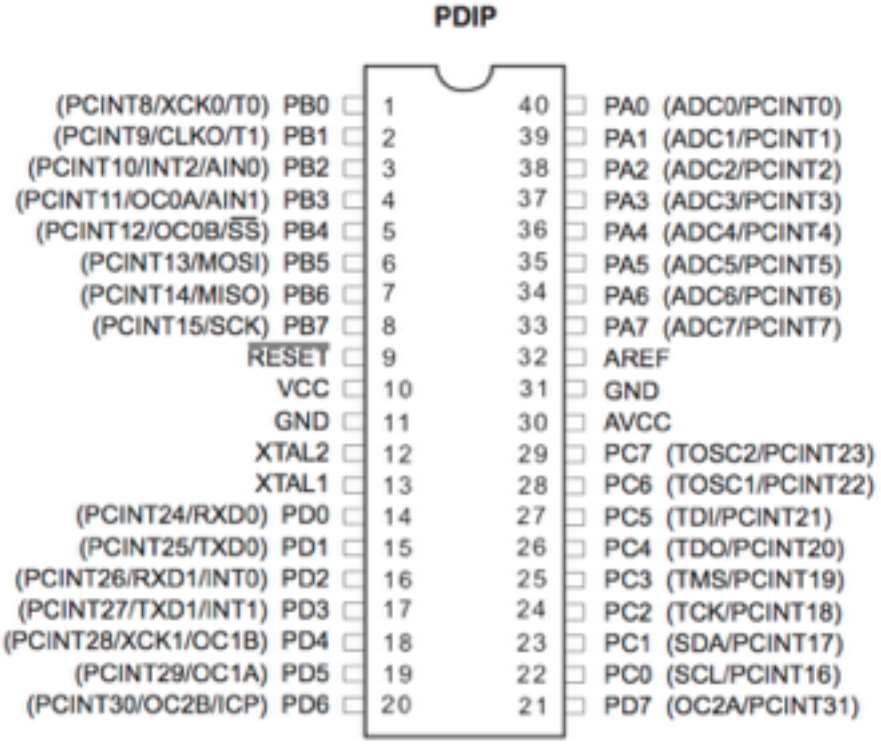
- Έναν μικροελεγκτή Atmel ATmega 644p στα 16 MHz
- Ένα bluetooth module hc-06, πρωτοκόλλου 1.1 με εμβέλεια 10 μέτρα
- Ένα ρελέ με γραμμή ελέγχου στα 5 volt
- Ένα τροφοδοτικό τύπου stepdown buck
- Ένας capacitive αισθητήρας αφής

2.1 Χαρακτηριστικά Μικροελεγκτή ATmega 644p

- Υψηλής απόδοσης και χαμηλής κατανάλωσης 8-bit μικροελεγκτής
- Προηγμένη αρχιτεκτονική RISC
  - 131 ισχυρές εντολές - Οι περισσότερες εκτελούνται σε έναν κύκλο
  - 32 x 8 Γενικού σκοπού καταχωρητές
  - Πλήρη στατική λειτουργία
  - Διακίνηση μέχρι 20 MIPS στα 20 MHz
  - Ενσωματωμένος πολλαπλασιαστής δύο κύκλων
- Υψηλής αντοχής - μη πτητικά τμήματα μνήμης
  - 64K bytes αυτοπρογραμματιζόμενη flash μνήμη
  - 2K bytes EEPROM
  - 4K bytes ROM
  - 10.000 κύκλοι εγγραφής-σβησίματος flash μνήμης
  - 100.000 κύκλοι εγγραφής-σβησίματος EEPROM

- Διατήρηση δεδομένων : 20 χρόνια σε θερμοκρασία 85 βαθμών κελσίου / 100 χρόνια σε 25 βαθμούς κελσίου
- Προαιρετικό διαμέρισμα κώδικα εκκίνησης με ανεξάρτητα Lock Bits
  - Προγραμματισμός μέσω ICSP
  - Δυνατότητα ανάγνωσης κατά την διάρκεια εγγραφής
- Κλείδωμα προγράμματος για ασφάλεια λογισμικού
- Διασύνδεση JTAG
  - Δυνατότητα σάρωσης σύμφωνα με το JTAG πρότυπο
  - Εκτενής υποστήριξη αποσφαλματοποίησης On-Chip
  - Προγραμματισμός της flash, EEPROM, ασφαλειών και bit κλειδώματος μέσω JTAG
- Περιφερειακές δυνατότητες
  - Δύο αριθμητές/μετρητές 8-bit με ξεχωριστούς προκλιμακωτές και λειτουργίες σύγκρισης
  - Ένας αριθμητής/μετρητής 16-bit με ξεχωριστούς προκλιμακωτές και λειτουργίες σύγκρισης
  - Μετρητής πραγματικού χρόνου με ξεχωριστό ταλαντωτή
  - Έξι κανάλια PWM
  - Μετατροπέας αναλογικού σήματος σε ψηφιακό με 8 κανάλια και 10-bit
    - Λειτουργία διαφόρισης με επιλέξιμο κέρδος(1x, 10x, 200x)
  - Σειριακή διασύνδεση δύο καλωδίων Byte-oriented
  - Δύο προγραμματιζόμενες σειριακές USART
  - Master/Slave σειριακή διασύνδεση SPI
  - Προγραμματιζόμενος μετρητής Watchdog με δύο ξεχωριστούς On-chip ταλαντωτές
  - On-chip αναλογικός συγκριτής
  - Διακοπή και αφύπνιση σε αλλαγή κατάστασης pin
- Ειδικές δυνατότητες Μικροελεγκτή
  - Ενεργοποίηση μετά από επαναφορά και προγραμματιζόμενη ανίχνευση Brown-out
  - Ενσωματωμένος βαθμονομημένος ταλαντωτής RC
  - Εξωτερικές και εσωτερικές πηγές διακοπών

- Έξι λειτουργίες αδράνειας : Idle, ADC Noise Reduction, Power-save, Power-down, Stand-by and Extended Standby
- Είσοδοι/Έξοδοι και συσκευασίες
  - 32 προγραμματιζόμενοι Είσοδοι/Έξοδοι
  - 40-pin PDIP, 44-lead TQFP, 44-pad VQFN/QFN/MLF (ATmega 644P)
- Λειτουργικές τάσεις
  - 2.7V - 5.5V
- Βαθμοί ταχύτητας
  - ATmega 644P : 0 - 10 MHz @ 2.7V - 5.5V, 0 - 20 MHz @ 4.5V - 5.5V
- Κατανάλωση Ισχύος στα 1 MHz, 1.8V, 25 βαθμούς κελσίου
  - Ενεργό : 0.4 mA
  - Λειτουργία power-down : 0.1μA
  - Λειτουργία power-save : 0.6 μA



Εικόνα 2.2 : Κατανομή των pins του μικροελεγκτή ATmega 644P

2.1.1 Περιγραφή των Pins του μικροελεγκτή

VCC

Ψηφιακή παροχή τάσης

GND

Γείωση

PORT A (PA7:PA0)

Λειτουργούν ως αναλογικοί είσοδοι για τον αναλογικό σε ψηφιακό μετατροπέα. Λειτουργούν επίσης σαν αμφίδρομες πόρτες εισόδου/εξόδου με εσωτερικούς αντιστάτες pull-up των 8-bit.

Οι buffer εξόδου έχουν χαρακτηριστικά συμμετρικής οδήγησης με υψηλές δυνατότητες πηγής.

Ως εισόδους τα pins της πόρτας A είναι εξωτερικά γειωμένα, καταναλώνουν ρεύμα εάν οι αντιστάτες pull-up είναι ενεργοί.
Τα pins της πόρτας A έχουν τρείς καταστάσεις, όταν ενεργοποιηθεί η κατάσταση επαναφοράς, ακόμα και αν δεν λειτουργεί.
Επίσης χρησιμοποιείται για διάφορες ειδικές λειτουργίες του ATmega 644P

PORT B (PB7:PB0)

Λειτουργούν σαν αμφίδρομες πόρτες εισόδου/εξόδου με εσωτερικούς αντιστάτες pull-up των 8-bit.
Οι buffer εξόδου έχουν χαρακτηριστικά συμμετρικής οδήγησης με υψηλές δυνατότητες πηγής.
Ως εισόδους τα pins της πόρτας B είναι εξωτερικά γειωμένα, καταναλώνουν ρεύμα εάν οι αντιστάτες pull-up είναι ενεργοί.
Τα pins της πόρτας B έχουν τρείς καταστάσεις, όταν ενεργοποιηθεί η κατάσταση επαναφοράς, ακόμα και αν δεν λειτουργεί.
Επίσης χρησιμοποιείται για διάφορες ειδικές λειτουργίες του ATmega 644P

PORT C (PC7:PC0)

Λειτουργούν σαν αμφίδρομες πόρτες εισόδου/εξόδου με εσωτερικούς αντιστάτες pull-up των 8-bit.
Οι buffer εξόδου έχουν χαρακτηριστικά συμμετρικής οδήγησης με υψηλές δυνατότητες πηγής.
Ως εισόδους τα pins της πόρτας B είναι εξωτερικά γειωμένα, καταναλώνουν ρεύμα εάν οι αντιστάτες pull-up είναι ενεργοί.
Τα pins της πόρτας B έχουν τρείς καταστάσεις, όταν ενεργοποιηθεί η κατάσταση επαναφοράς, ακόμα και αν δεν λειτουργεί.
Επίσης χρησιμοποιείται για τις λειτουργίες της διασύνδεσης JTAG καθώς και για διάφορες ειδικές λειτουργίες του ATmega 644P

PORT D (PD7:PD0)

Λειτουργούν σαν αμφίδρομες πόρτες εισόδου/εξόδου με εσωτερικούς αντιστάτες pull-up των 8-bit.

Οι buffer εξόδου έχουν χαρακτηριστικά συμμετρικής οδήγησης με υψηλές δυνατότητες πηγής.
Ως εισόδους τα pins της πόρτας Β είναι εξωτερικά γειωμένα, καταναλώνουν ρεύμα εάν οι αντιστάτες pull-up είναι ενεργοί.
Τα pins της πόρτας Β έχουν τρείς καταστάσεις, όταν ενεργοποιηθεί η κατάσταση επαναφοράς, ακόμα και αν δεν λειτουργεί.
Επίσης χρησιμοποιείται για διάφορες ειδικές λειτουργίες του ATmega 644P

RESET

Είσοδος επαναφοράς. Μια χαμηλή τάση σε αυτό το pin για περισσότερο από το ελάχιστο μήκος παλμού θα τροφοδοτήσει την επαναφορά, ακόμα και αν το ρολόι δεν λειτουργεί. Οι παλμοί που έχουν μήκος μικρότερο του ελαχίστου παλμού δεν εγγυώνται την επαναφορά.

XTAL1

Είσοδος του αναστρέφοντος ενισχυτή ταλάντωσης και του κυκλώματος λειτουργίας εσωτερικού ρολογιού.

XTAL2

Έξοδος του αναστρέφοντος ενισχυτή ταλάντωσης

AVCC

Είναι η τροφοδοσία για τον μετατροπέα αναλογικού σήματος σε ψηφιακό της πόρτας Α. Θα πρέπει να συνδέεται εξωτερικά στην τροφοδοσία ακόμα και αν δεν χρησιμοποιείται ο μετατροπέας αναλογικού σήματος σε ψηφιακό. Εάν χρησιμοποιείται ο μετατροπέας αναλογικού σήματος σε ψηφιακό, θα πρέπει να συνδέεται στην τροφοδοσία μέσω ενός χαμηλοπερατού φίλτρου.

AREF

Το pin αναλογικής αναφοράς για τον μετατροπέα αναλογικού σήματος σε ψηφιακό

[Πηγή: 1]

2.2 Arduino Boot Loader

- Boot Loader είναι :

Οι μικροελεγκτές συνήθως προγραμματίζονται μέσω ενός προγραμματιστή εκτός και εάν έχουν ένα κομμάτι κώδικα, το οποίο επιτρέπει την εγκατάσταση νέων προγραμμάτων με την χρήση εξωτερικού προγραμματιστή. Αυτό ονομάζεται Boot Loader

- Αν δεν θες να χρησιμοποιήσεις Boot Loader

Εάν είναι απαραίτητη η χρήση του πλήρους αποθηκευτικού χώρου προγραμμάτων του μικροελεγκτή ή για να αποφευχθεί η καθυστέρηση που προκαλεί ο Boot Loader, μπορούμε να γράψουμε τα προγράμματα μας με την χρήση εξωτερικού προγραμματιστή

- Εγγραφή Boot Loader

Για την εγγραφή του Boot Loader σε άδεια AVR chip, χρησιμοποιήσαμε ένα arduino mega στο οποίο συνδέσαμε τα ICSP pin του arduino mega στα ICSP pin του άδειου AVR chip. Στην συνέχεια επιλέξαμε από την διεπαφή του προγράμματος arduino το υπό-μενού tools->burn boot loader ενώ στη συνέχεια προκαλέσαμε επαναφορά στο arduino mega έτσι ώστε να αναγνωρίσει το συνδεδεμένο άδειο chip.

- Πώς λειτουργεί ο Boot Loader

Η εντολή “Burn Boot Loader” στην διεπαφή arduino χρησιμοποιεί ένα εργαλείο ανοιχτού κώδικα το avrdude. Πραγματοποιούνται τέσσερα βήματα :

1. Ξεκλείδωμα του τμήματος boot loader στο chip
2. Ρύθμιση των ασφαλειών στο chip
3. Εγγραφή του κώδικα boot loader στο chip
4. Κλείδωμα του τμήματος boot loader στο chip

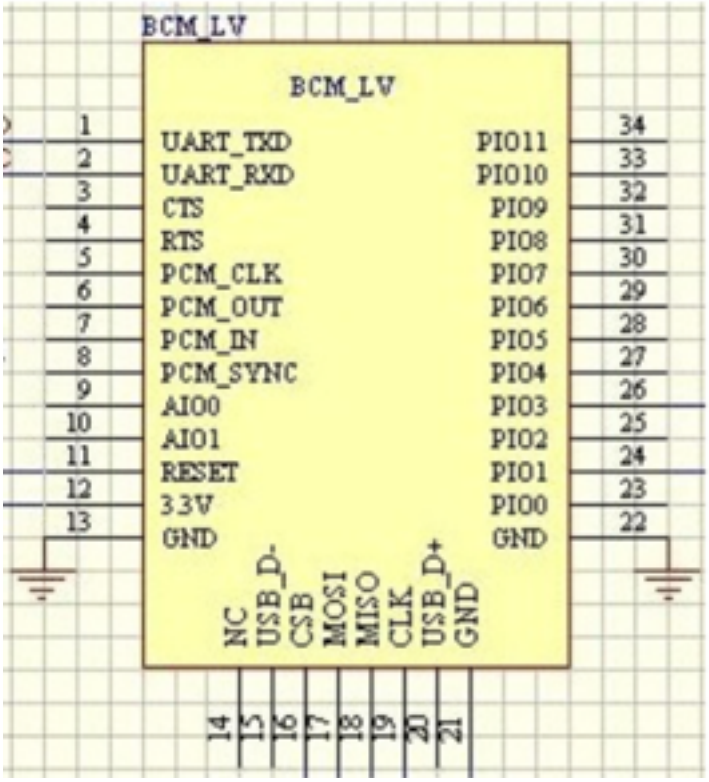
2.3 Προγραμματισμός ATmega 644P

Για τον προγραμματισμό του Arduino ATmega 644p χρησιμοποιήσαμε την βιβλιοθήκη Sanguino η οποία περιέχει οδηγίες προς το εργαλείο avrdude προκειμένου να επιτρέψει την εγγραφή δεδομένων προγράμματος μέσω του Arduino Mega στο Atmega 644p. Η βιβλιοθήκη Sanguino είναι μια βιβλιοθήκη η οποία επιτρέπει τον προγραμματισμό μικροελεγκτών ATmega 644/1284 μέσω της διεπαφής Arduino. Περιέχει οδηγίες για την τοποθεσία διευθύνσεων μνήμης και τον προγραμματισμό ασφαλειών, προκειμένου να μπορούν να εγγραφούν προγράμματα στην μνήμη του μικροελεγκτή. Για τον προγραμματισμό του μικροελεγκτή επιλέγουμε από την διεπαφή Arduino το μενού Arduino->Preferences και στην συνέχεια εισάγουμε το URL “https://raw.githubusercontent.com/Lauszus/Sanguino/master/package_lauszus_sanguino_index.json” στο πεδίο “Additional Board Manager URLs” και επιλέγουμε το OK. Στην συνέχεια επιλέγουμε το μενού Tools->Boards->Board Manager και επιλέγουμε το Sanguino by Kristian Lauth και επιλέγουμε το install. Τέλος αρκεί να επιλέξουμε ως board το Sanguino ενώ σαν processor το ATmega 644p 16MHz και να πλοηγηθούμε στο μενού Sketch και να επιλέξουμε το Upload Using Programmer. Τα ICSP pins του Arduino Mega θα πρέπει να είναι συνδεδεμένα στα αντίστοιχα ICSP pin του ATmega 644p.

2.4 Bluetooth Module HC-06

- Ασύρματος πομποδέκτης
  - Ευαισθησία εώς(Bit error rate) -80dBm.
  - Ρυθμός μεταβολής ισχύος εξόδου: -4 - +6dBm.
- Function description (perfect Bluetooth solution)
- EDR Module με εύρος μεταβολής διαμόρφωσης βάθους : 2Mbps - 3Mbps.
- Ενσωματωμένη κεραία 2.4GHz
- Λειτουργική τάση: 3.1V~4.2V. Ένταση ρεύματος σε λειτουργία σύζευξης: 30~40mA.
- Ένταση ρεύματος σε λειτουργία επικοινωνίας: 8mA.
- Standard θύρα HCI (UART ή USB)
- Πρωτόκολλο USB: Full Speed USB1.1, Συμβατό με USB2.0

- Οικολογικά συμβατό με RoHS πιστοποίηση.
- Περιέχει πομποδέκτη 2.4 GHz
- Βασισμένο σε τεχνολογία Bluetooth CSR BC04
- Λειτουργία προσαρμοζόμενης αναπήδησης συχνότητας
- Κατανάλωση ενέργειας Bluetooth τάξης 2
- Εύρος θερμοκρασίας αδράνειας:-40 °C - 85°C,εύρος θερμοκρασίας λειτουργίας: -25 °C - +75°C
- Ισχύς μετάδοσης: 3 dBm



Εικόνα 2.3 : Κατανομή των pins του Bluetooth module HC-06

2.4.1 Περιγραφή των PINS

GND	Γείωση
1V8	1.8V τροφοδοσία με γραμμικό ρυθμιστή εξόδου εύρους 1.7-1.9V
VCC	τροφοδοσία

AIOX	Αμφίδρομη προγραμματιζόμενη είσοδος/έξοδος
PIO0	Αμφίδρομη προγραμματιζόμενη είσοδος/έξοδος και γραμμή εξόδου της LNA
PIO1	Αμφίδρομη προγραμματιζόμενη είσοδος/έξοδος και γραμμή εξόδου της PA
PIO2-11	Αμφίδρομη προγραμματιζόμενη είσοδος/έξοδος
RESETB	Προκαλεί επαναφορά στο chip
UART-RTS	Αίτηση αποστολής προς το UART, ενεργοποίηση με λογικό 0
UART-CTS	Έγκριση αποστολής προς το UART, ενεργοποίηση με λογικό 0
UART_RX	Είσοδος δεδομένων
UART_TX	Έξοδος δεδομένων
SPI_MOSI	Είσοδος δεδομένων σειριακής περιφερειακής διεπαφής
SPI_CSB	Επιλογή SPI διεπαφής, ενεργοποιείται με λογικό 0
SPI_CLK	Ρολόι σειριακής σε περιφερειακή διεπαφή
SPI_MISO	Έξοδος δεδομένων σειριακής περιφερειακής διεπαφής
USB-	
USB+	
1.8V	
PCM_CLK	
PCM_OUT	
PCM_IN	
PCM_SYNC	

Πίνακας 2.1 : Περιγραφή των pins του bluetooth module HC-06

[Πηγή:2]

2.5 Ρελέ

Το ρελέ είναι ένας ηλεκτρικά ελεγχόμενος διακόπτης. Τα περισσότερα ρελέ χρησιμοποιούν ένα ηλεκτρομαγνήτη για να λειτουργήσουν μηχανικά έναν διακόπτη, αλλά χρησιμοποιούνται και άλλες μέθοδοι, όπως τα σταθερής κατάστασης ρελέ. Τα

ρελέ χρησιμοποιούνται όταν είναι απαραίτητο να ελέγξεις ένα κύκλωμα με ένα σήμα χαμηλής ισχύος(με πλήρη ηλεκτρική απομόνωση μεταξύ του κυκλώματος ελέγχου και του ελεγχομένου κυκλώματος), ή όταν μερικά κυκλώματα πρέπει να ελεγχθούν από ένα σήμα. Τα πρώτα ρελέ χρησιμοποιήθηκαν σε κυκλώματα τηλεγράφου μεγάλης απόστασης ως ενισχυτές: επαναλάμβαναν το σήμα που ερχόταν από ένα κύκλωμα το αναμετέδιδαν σε ένα άλλο κύκλωμα. Τα ρελέ χρησιμοποιούνταν εκτενώς σε τηλεπικοινωνίες και στους πρώτους υπολογιστές για να εκτελέσουν λογικές πράξεις.

Ένας τύπος ρελέ που μπορεί να διαχειριστεί την υψηλή τάση που απαιτείται για τον έλεγχο ενός ηλεκτρικού κινητήρα ή άλλων φορτίων ονομάζεται contactor. Τα σταθερής κατάστασης ρελέ ελέγχουν κυκλώματα ισχύος χωρίς κινητά μέρη, αντίθετα χρησιμοποιείται μια συσκευή ημιαγωγού για να λειτουργήσει σαν διακόπτης. Τα ρελέ με βαθμονομημένα χαρακτηριστικά λειτουργίας αλλά και με πολλαπλά πηνία λειτουργίας χρησιμοποιούνται για να προστατέψουν ηλεκτρικά κυκλώματα από υπερφόρτωση ή σφάλματα. Σε σύγχρονα ηλεκτρικά συστήματα ισχύος αυτές οι λειτουργίες εκτελούνται από ψηφιακά όργανα τα οποία ονομάζονται ρελέ προστασίας.

Τα μαγνητικά ρελέ απαιτούν ένα σήμα ισχύος πηνίου για να μετακινήσουν τις επαφές σε μία κατεύθυνση και άλλον έναν ανάστροφο παλμό για να μετακινηθούν οι επαφές στην αρχική τους θέση. Οι επαναλαμβανόμενοι παλμοί από την ίδια είσοδο δεν επηρεάζουν τις επαφές. Τα μαγνητικά ρελέ χρησιμεύουν σε εφαρμογές όπου διακοπή στην τάση του πηνίου δεν πρέπει να αλλάξει την τάση στις επαφές.

Τα μαγνητικά ρελέ μπορεί να έχουν μονά ή διπλά πηνία. Σε συσκευή με μονό πηνίο, το ρελέ θα λειτουργήσει προς μια κατεύθυνση όταν εφαρμοστεί ισχύς σε μία πολικότητα, και θα επαναφερθεί όταν η πολικότητα αντιστραφεί. Σε συσκευή με διπλό πηνίο όταν πολωμένη τάση εφαρμοστεί στο πηνίο επαναφοράς οι επαφές θα αλλάξουν.

2.5.1 Βασική λειτουργία και σχεδίαση

Ένα απλό ηλεκτρομαγνητικό ρελέ αποτελείται από ένα καλώδιο τυλιγμένο γύρω από έναν πυρήνα μαλακού σιδήρου, ένα κινητός οπλισμός σιδήρου, και ένα ή περισσότερα σετ επαφών (υπάρχουν δύο επαφές στο ρελέ που απεικονίζεται).

Συγκρατείται στη θέση του από ένα ελατήριο, έτσι ώστε όταν το ρελέ απενεργοποιείται υπάρχει ένα κενό αέρα στο μαγνητικό κύκλωμα. Σε αυτή την κατάσταση, ένα από τα δύο σύνολα επαφών στο ρελέ που απεικονίζεται είναι κλειστό, και η άλλη ομάδα επαφών είναι ανοιχτή. Άλλα ρελέ μπορούν να έχουν περισσότερα ή λιγότερα σετ επαφών ανάλογα με τη λειτουργία τους.

Όταν το ηλεκτρικό ρεύμα διέρχεται μέσα από το πηνίο παράγει ένα μαγνητικό πεδίο που ενεργοποιεί τον οπλισμό, και η μετακίνηση της κινούμενης επαφής είτε δημιουργεί είτε διακόπτει μια σύνδεση με μια σταθερή επαφή. Αν το σύνολο των επαφών ήταν κλειστό όταν το ρελέ απενεργοποιείται, τότε η κίνηση ανοίγει τις επαφές και διακόπτει τη σύνδεση, και το αντίστροφο, αν οι επαφές ήταν ανοικτές. Όταν το ρεύμα στο πηνίο είναι απενεργοποιημένο, ο οπλισμός επιστρέφεται από μια δύναμη, περίπου κατά το ήμισυ ισχυρή με την μαγνητική δύναμη, στη αρχική του θέση. Συνήθως αυτή η δύναμη παρέχεται από ένα ελατήριο, αλλά η βαρύτητα χρησιμοποιείται επίσης ευρέως σε βιομηχανικούς εκκινητές μηχανών. Τα περισσότερα ρελέ κατασκευάζονται για να λειτουργούν γρήγορα. Σε μια περίπτωση χαμηλής τάσης αυτό επιτρέπει την μείωση θορύβου. Σε περίπτωση υψηλής τάσης ή ρεύματος επιτρέπει την μειώση τόξου.

Όταν το πηνίο ενεργοποιείται με συνεχές ρεύμα, μία δίοδος τοποθετείται συχνά στα άκρα του πηνίου για να διαχέει την ενέργεια από το μαγνητικό πεδίο στην απενεργοποίηση, η οποία διαφορετικά θα δημιουργούσε μια υψηλή τάση επικίνδυνη για ένα κυκλώμα ημιαγωγών. Τέτοιες δίοδοι δεν χρησιμοποιήθηκαν ευρέως πριν από την εφαρμογή των τρανζίστορ ως ελεγκτές ρελέ, αλλά σύντομα έγινε συνηθισμένη διαδικασία καθώς τα τρανζίστορ γερμάνιου καταστρέφονταν εύκολα από αυτό το φαινόμενο. Μερικά ρελέ αυτοκινήτων περιλαμβάνουν μια δίοδο στο εσωτερικό της θήκης του ρελέ.

Εάν το ρελέ οδηγεί ένα μεγάλο, ή σημαντικότερα ένα δραστικό φορτίο, μπορεί να υπάρχει ένα παρόμοιο πρόβλημα ρευμάτων με μεγάλη αυξομείωση της τάσης γύρω από τις επαφές εξόδου του ρελέ. Στην περίπτωση αυτή, ένα κύκλωμα αποσβεστήρα (ενός πυκνωτή και μιας αντίστασης σε σειρά) μεταξύ των επαφών μπορεί να απορροφήσει το κύμα.

Εάν το πηνίο έχει σχεδιαστεί για να ενεργοποιείται με εναλλασσόμενο ρεύμα (AC), κάποια μέθοδος χρησιμοποιείται για να χωρίσει την ροή σε δύο εκτός-φάσης συστατικά τα οποία προστηθέμενα μαζί, αυξάνουν την ελάχιστη έλξη στον οπλισμό

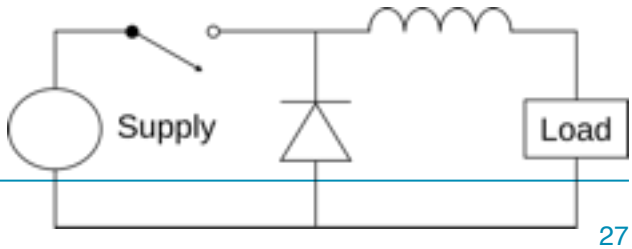
κατά τη διάρκεια του κύκλου AC. Συνήθως αυτό γίνεται με ένα μικρό σύρμα χαλκού τοποθετημένο γύρω από ένα τμήμα του πυρήνα που δημιουργεί την καθυστερημένη, εκτός φάσης συνιστώσα του εναλλασσόμενου ρεύματος, η οποία κρατάει τις επαφές σταθερές κατά τη διάρκεια των διελεύσεων μηδενικού ρεύματος τάσης ελέγχου.
[Πηγή: 3]

2.6 Τροφοδοτικό τύπου Step down buck

Εικόνα 2.4 : Step down buck converter

Ένας buck converter είναι ένας dc to dc μετατροπέας ισχύος ο οποίος μειώνει την τάση που δέχεται στην είσοδο(ενώ αυξάνει το ρεύμα). Είναι ένα τροφοδοτικό τύπου switched-mode το οποίο συνήθως αποτελείται από τουλάχιστον δύο ημιαγωγούς(μία δίοδο και ένα τρανζίστορ, αν και οι σύγχρονοι μετατροπείς buck αντικαθιστούν την δίοδο με ένα δεύτερο τρανζίστορ το οποίο χρησιμοποιείται για σύγχρονη ανόρθωση) και τουλάχιστον ένα στοιχείο για αποθήκευση ενέργειας, έναν πυκνωτή, ένας επαγωγέας, ή και τα δύο σε συνδυασμό. Για να μειωθεί η τάση προστίθενται φίλτρα που αποτελούνται από πυκνωτές(κάποιες φορές σε συνδυασμό με επαγωγείς) που συνήθως τοποθετούνται στην έξοδο και την είσοδο τέτοιων μετατροπέων.

Οι μετατροπείς dc to dc προσφέρουν πολύ καλύτερη απόδοση σε σύγκριση με τους γραμμικούς



ρυθμιστές, οι οποίοι έχουν πιο απλό κύκλωμα που μειώνει την τάση με την απόθεση

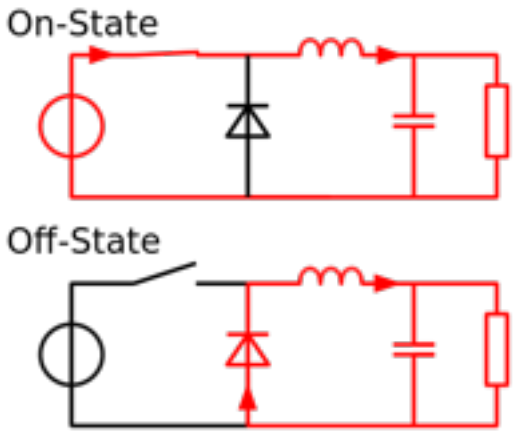
Εικόνα 2.5 : Διάγραμμα κυκλώματος του step down buck converter

ισχύος σαν θερμότητα το οποίο δεν ενισχύει το ρεύμα εξόδου.

Οι μετατροπείς buck μπορεί έχουν εκπληκτική απόδοση(συνήθως μεγαλύτερη από 90%), που τους κάνει χρήσιμους σε εφαρμογές όπως η μετατροπή της κύριας τάσης τροφοδοσίας ενός υπολογιστή(συνήθως 12V) σε χαμηλότερες τάσης που απαιτούνται από το USB, την DRAM, και τον επεξεργαστή.

2.6.1 Θεωρία λειτουργίας

Η βασική αρχή λειτουργίας του μετατροπέα buck τοποθετεί το ρεύμα σε έναν επαγωγέα που ελέγχεται από δύο διακόπτες(συνήθως ένα τρανζίστορ και μία δίοδο).

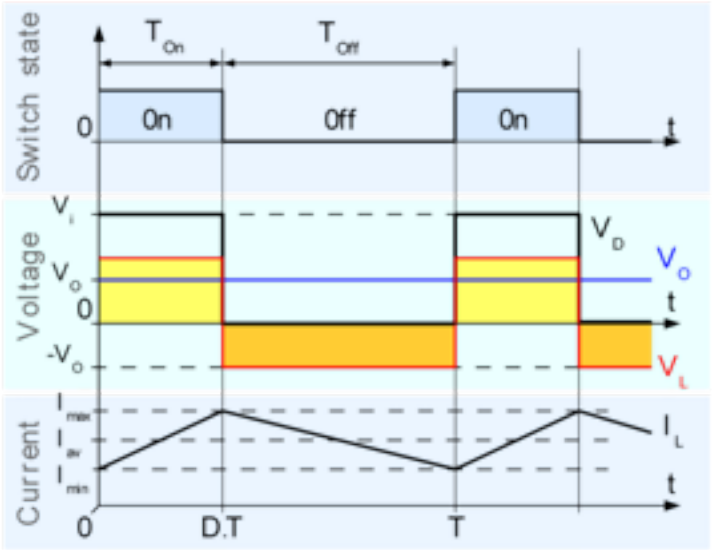


Εικόνα 2.6 : Διάγραμμα καταστάσεων του step down buck converter

Στον ιδανικό μετατροπέα, όλα τα εξαρτήματα θεωρούμε ότι είναι τέλεια. Συγκεκριμένα ο διακόπτης και η δίοδος έχουν μηδενική πτώση τάσης όταν είναι ανοιχτά και μηδενική ροή ρεύματος όταν είναι απενεργοποιημένα και ο επαγωγέας έχει μηδέν αντίσταση σε σειρά. Επιπλέον υποθέτουμε ότι οι τάσεις εισόδου και εξόδου δεν διαφοροποιούνται.

2.6.2 Σκεπτικό

Το σκεπτικό του μετατροπέα buck μπορούμε να το αντιληφθούμε καλύτερα με τους όρους της σχέση μεταξύ ρεύματος και τάσης του επαγωγέα. Ξεκινώντας με τον διακόπτη ανοιχτό , το ρεύμα στο κύκλωμα είναι μηδέν. Όταν ο διακόπτης είναι αρχικά κλειστός το ρεύμα αρχίσει να αυξάνεται , και ο επαγωγέας θα παράξει μια αντίστροφη τάση ανάμεσα στις επαφές του ως αντίδραση στην αλλαγή τάσης. Αυτή η πτώση τάσης αντιδρά στην τάση της πηγής και συνεπώς μειώνει την τελική τάση στο φορτίο. Κατά την διάρκεια του χρόνου ο ρυθμός αλλαγής του ρεύματος μειώνεται, και η τάση στον επαγωγέα επίσης μειώνεται, αυξάνοντας την τάση στο φορτίο. Κατά το διάστημα αυτό , ο επαγωγέας αποθηκεύει ενέργεια με την μορφή ενός μαγνητικού πεδίου. Εάν ο διακόπτης ανοίξει καθώς το ρεύμα μεταβάλλεται, τότε πάντα θα υπάρχει μια πτώση τάσης στον επαγωγέα, συνεπώς η τελική τάση στο φορτίο θα



Εικόνα 2.7 : Αλλαγές τάσεων και ρευμάτων κατά την διάρκεια συνεχής λειτουργίας ενός ιδανικού step down buck converter

είναι πάντα μικρότερη από την τάση εισόδου. Όταν ο διακόπτης μεταβληθεί ξανά, η τάση εισόδου θα αφαιρεθεί από το κύκλωμα και το ρεύμα θα μειωθεί. Το ρεύμα που μεταβάλλεται θα δημιουργήσει μια αλλαγή στην τάση στον επαγωγέα, ο οποίος θα μετατραπεί σε πηγή τάσης. Η αποθηκευμένη ενέργεια στο μαγνητικό πεδίο του επαγωγέα υποστηρίζει την ροή ρεύματος προς το φορτίο. Στο διάστημ αυτό, ο επαγωγέας αποφορτίζει την αποηθηκευμένη ενέργεια στο υπόλοιπο κύκλωμα. Εάν ο

διακόπτης κλείσει ξανά πριν ο επαγωγέας αποφορτιστεί πλήρως, η τάση στο φορτίο θα είναι πάντα μεγαλύτερη από μηδέν.

Το τροφοδοτικό το οποίο χρησιμοποιήθηκε για αυτή την εργασία είναι ένα τροφοδοτικό το οποίο χρησιμοποιεί έναν AC/DC buck converter, του οποίου οι αρχές λειτουργίας είναι ίδιες με τον DC/DC buck converter με την διαφορά ότι στην είσοδο του υπάρχει μια γέφυρα ανόρθωσης η οποία ανορθώνει το AC σήμα σε DC. Η μετατροπή που πραγματοποιεί η συγκεκριμένη γέφυρα αναλύεται παρακάτω.

Μια γέφυρα διόδων αποτελείται από 4 ή περισσότερες διόδους σε μια διαμόρφωση κυκλώματος γέφυρας που παρέχει την ίδια πολικότητα στην έξοδο όπως η

Εικόνα 2.8 : Γέφυρα ανόρθωσης

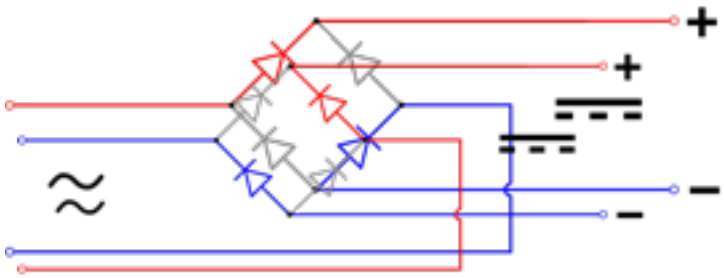
πολικότητα της εισόδου. Όταν χρησιμοποιείται στην πιο συνηθισμένη του εφαρμογή, για την μετατροπή μιας εισόδου εναλλασσόμενου ρεύματος σε μια έξοδο συνεχούς ρεύματος, είναι γνωστός και ως ανορθωτής γέφυρας. Ένας ανορθωτής γέφυρας παρέχει ανόρθωση πλήρους κύματος από μια πηγή AC 2 καλωδίων, που αποτελεί μια πιο φθηνή και ελαφριά λύση σε σχέση με έναν ανορθωτή με είσοδο 3 καλωδίων από έναν μετασχηματιστή.

Το βασικό στοιχείο μια γέφυρας διόδων είναι ότι η πολικότητα της εξόδου είναι ίδια με τη πολικότητα στην είσοδο χωρίς να μας ενδιαφέρει ποια είναι αυτή. Το κύκλωμα γέφυρας διόδων ήταν εφεύρεση του πολωνού ηλεκτροτεχνικού Karol Pollak και κατοχυρώθηκε με δίπλωμα ευρεσιτεχνίας στις 14 Ιαν, 1896.

2.6.3 Βασική Λειτουργία

Σύμφωνα με το συμβατικό μοντέλο ροής του ρεύματος, το ρεύμα ορίζεται ως θετικό όταν ρέει μέσα από ηλεκτρικούς αγωγούς από τον θετικό στον αρνητικό πόλο. Στην πραγματικότητα, τα ελεύθερα ηλεκτρόνια μέσα σε έναν αγωγό σχεδόν πάντα ρέουν από τον αρνητικό στον θετικό πόλο. Στις περισσότερες εφαρμογές, όμως, η πραγματική κατεύθυνση της ροής ρεύματος δεν μας ενδιαφέρει. Συνεπώς, στην παρακάτω συζήτηση χρησιμοποιείται η συμβατική μέθοδος.

Στα παρακάτω διαγράμματα όταν η είσοδος που έχει συνδεθεί στην αριστερή γωνία του ρόμβου είναι θετική και η είσοδος που έχει συνδεθεί στη δεξιά γωνία είναι



αρνητική, το ρεύμα ρέει από το επάνω τερματικό προς τα δεξιά κατά μήκος του κόκκινου μονοπατιού και προς την έξοδο και επιστρέφει στο κάτω τερματικό άκρο μέσω της μπλε διαδρομής.

Όταν η είσοδος που έχει συνδεθεί στην αριστερή γωνία του ρόμβου είναι αρνητική και η είσοδος που έχει συνδεθεί στη δεξιά γωνία είναι θετική, το ρεύμα ρέει από το

Εικόνα 2.9 : Γέφυρα ανόρθωσης

κάτω τερματικό προς τα δεξιά κατά μήκος του κόκκινου μονοπατιού και προς την έξοδο και επιστρέφει στο επάνω τερματικό άκρο μέσω της μπλε διαδρομής.

Σε κάθε περίπτωση, η επάνω δεξιά έξοδος παραμένει θετική και η κάτω δεξιά έξοδος παραμένει αρνητική. Αφού αυτό είναι αληθές είτε η είσοδος είναι AC είτε είναι DC, αυτό το κύκλωμα όχι μόνο παράγει μια DC έξοδο από μια AC είδοσο, μπορεί επίσει να παρέχει αυτό που αποκαλείται “προστασία ανάστροφης πολικότητας”. Αυτό είναι, η δυνατότητα να λειτουργεί ένα DC κύκλωμα ακόμα και αν η DC τροφοδοσία έχει αντιστραφεί, όπως συμβαίνει όταν πολώνουμε ανάποδα μια μπαταρία, και να προστατεύει το κύκλωμα από πιθανή ζημιά που προκαλείται από ανάποδη πολικότητα.

Πριν την διάθεση ολοκληρωμένων κυκλωμάτων, ένας ανορθωτής γέφυρας κατασκευαζόταν από διακριτά εξαρτήματα, δλδ. ξεχωριστές διόδους. Από το 1950 ένα εξάρτημα 4 ακίδων που εμπεριέχει 4 διόδους συνδεδεμένο σε συνδεσμολογία γέφυρας έγινε καθιερωμένο εμπορικό εξάρτημα και τώρα είναι διαθέσιμο σε διάφορες κατηγορίες τάσης και ρεύματος.

2.6.4 Εξομάλυνση εξόδου

Για πολλές εφαρμογές, ειδικά για εναλλασσόμενο ρεύμα μονής φάσης όπου η γέφυρα πλήρους κύματος χρησιμεύει στην μετατροπή της εναλλασσόμενης εισόδου σε συνεχή, η προσθήκη ενός πυκνωτή μπορεί να είναι επιθυμητή γιατί η γέφυρα από μόνης της παρέχει μια παλμική έξοδο DC σήματος.

Η λειτουργία του πυκνωτή, γνωστός και ως πυκνωτής κατακράτησης είναι να μικρύνει την διαφορά(ή να ‘εξομαλύνει’) στην ανακτημένη AC κυματομορφή της τάσης εξόδου από την γέφυρα. Υπάρχει ακόμα μια διαφοροποίηση η οποία είναι γνωστή ως κυματισμός. Μια επεξήγηση της ‘εξομάλυνσης’ είναι ότι ο πυκνωτής παρέχει ένα μονοπάτι χαμηλής αντίστασης στο εναλλασσόμενο μέρος του σήματος εξόδου, μειώνοντας την εναλλασσόμενη τάση και το εναλλασσόμενο ρεύμα σε όλο το φορτίο. Σε λιγότερο τεχνικούς όρους, οποιαδήποτε πτώση στην τάση και στο ρεύμα εξόδου της γέφυρας τείνει να ακυρώνεται από τις απώλειες φορτίου στον πυκνωτή. Αυτό το φορτίο ρέει προς τα έξω σαν επιπλέον ρεύμα μέσα από το φορτίο. Συνεπώς η μεταβολή του ρεύματος και της τάσης φορτίου είναι μειωμένη σχετικά με το τι θα συνέβαινε χωρίς τον πυκνωτή, μειώνοντας έτσι τις αλλαγές στο ρεύμα/τάση της εξόδου.

[Πηγή: 4]

Κεφάλαιο 3

Homekit

3.1 Εισαγωγή

Με το Homekit, οι χρήστες μπορούν να χρησιμοποιούν εφαρμογές οικιακού αυτοματισμού στις iOS συσκευές για να ελέγχουν και να διαμορφώνουν τα δικτυωμένα αξεσουάρ στα σπίτια τους, ανεξαρτήτως κατασκευαστή. Συνήθως, μια εφαρμογή οικιακού αυτοματισμού πρέπει να βοηθάει τους χρήστες στις παρακάτω ενέργειες :

- Διαμόρφωση του σπιτιού
- Διαχείριση χρηστών
- Προσθήκη/Αφαίρεση αξεσουάρ
- Διαμόρφωση σκηνών

Επιπλέον, μια εφαρμογή οικιακού αυτοματισμού πρέπει να είναι εύκολη και απολαυστική στη χρήση. Μερικοί τρόποι για να δημιουργήσετε μια εξαιρετική εμπειρία :

- Ενσωμάτωση με τη Siri
- Αυτόματη εύρεση αξεσουάρ
- Χρήση κατάλληλης γλώσσας

3.2 Διαμόρφωση σπιτιού

Το σύστημα Homekit έχει δημιουργηθεί με βάση τρεις τύπους τοποθεσιών: Δωμάτια, Ζώνες και Σπίτια. Δωμάτια, όπως σαλόνι ή κρεβατοκάμαρα, αποτελούν τη βασική οργανωτική έννοια και μπορεί να περιέχουν έναν οποιονδήποτε αριθμό αξεσουάρ. Οι ζώνες είναι σύνολα δωματίων, όπως “ένας όροφος”.
Οι χρήστες πρέπει να ορίσουν τουλάχιστον ένα Σπίτι που θα χρησιμοποιηθεί ως τοποθεσία των αξεσουάρ. Κάθε Σπίτι περιέχει Δωμάτια και προαιρετικά και Ζώνες. Τα Δωμάτια και οι Ζώνες κάνουν εύκολη στους χρήστες, την εύρεση και τον έλεγχο των αξεσουάρ. Οι εφαρμογές πρέπει να παρέχουν τρόπους δημιουργίας, ονομασίας, μετατροπής και διαγραφής Σπιτιών, Δωματίων και Ζωνών.

Εάν κάποιος έχει πολλαπλά Σπίτια, η επιλογή για τον προσδιορισμό του προκαθορισμένου Σπιτιού, επιτρέπει γρηγορότερη διαμόρφωση των νέων αξεσουάρ.

3.3 Διαχείριση Χρηστών

Οι εφαρμογές πρέπει να παρέχουν τρόπους για την διαχείριση των χρηστών που επιτρέπεται να ελέγχουν τα αξεσουάρ σε ένα Σπίτι. Όταν ένας λογαριασμός iCloud προστίθεται σε ένα Σπίτι, ο κάτοχος του λογαριασμού μπορεί να επηρεάσει τα χαρακτηριστικά των αξεσουάρ. Όταν ένας κάτοχος λογαριασμού υποδεικνύεται ως διαχειριστής, μπορεί να προσθέσει νέα αξεσουάρ, να διαχειριστεί χρήστες, να διαμορφώσει Σπίτια.

3.4 Πρόσθεση και αφαίρεση αξεσουάρ

Είναι σημαντικό να γίνεται η προσθήκη νέων αξεσουάρ γρήγορα και εύκολα. Οι εφαρμογές θα πρέπει να αναζητούν αυτόματα για νέα αξεσουάρ και να τα παρουσιάζουν εμφανώς στο UI.

Οι χρήστες χρειάζονται έναν τρόπο για να αναγνωρίζουν το αντικείμενο που επηρεάζουν, οπότε θα πρέπει οι προγραμματιστές να παρέχουν έναν τρόπο γρήγορης ταυτοποίησης ενός αξεσουάρ. Για παράδειγμα σε μια λάμπα, μπορείς να επιτρέψεις στους χρήστες να την ανάψουν, προκειμένου να την ταυτοποιήσουν στο χώρο.

Η διαμόρφωση θα πρέπει να συμπεριλαμβάνει την ανάθεση ενός Ονόματος, Σπιτιού, Δωματίου και προαιρετικά μιας Ζώνης σε ένα αξεσουάρ. Οι διαχειριστές πρέπει να εισάγουν τον κωδικό διαμόρφωσης του αξεσουάρ(συμπεριλαμβάνεται στο εγχειρίδιο ή στην συσκευασία της συσκευής) για την ολοκλήρωση της διαδικασίας συσχέτισης σε ένα Σπίτι.

Το Apple Wireless Accessory Configuration (WAC) χρησιμοποιείται για την προσθήκη αξεσουάρ που υποστηρίζουν Wi-Fi σε ένα οικιακό δίκτυο. Οι χρήστες έχουν πρόσβαση στο WAC από τις ρυθμίσεις ή μέσα από εφαρμογές. Μετά την χρήση του WAC για την διαμόρφωση του αξεσουάρ, οι χρήστες μπορούν να το προσθέσουν σε ένα Σπίτι και να του αναθέσουν ένα Όνομα και ένα Δωμάτιο. Σημειώστε ότι θα πρέπει πάντα οι χρήστες να εκκινούν την εύρεση και διαμόρφωση των αξεσουάρ εκκινώντας την εφαρμογή.

3.5 Εύρεση αξεσουάρ

Θα πρέπει οι χρήστες να έχουν πρόσβαση σε πολλαπλούς τρόπους, να βρίσκουν γρήγορα αξεσουάρ. Η ώρα της ημέρας, η εποχή και η τοποθεσία του χρήστη μπορούν να επηρεάσουν ποια αξεσουάρ έχουν σημασία εκείνη την στιγμή, οπότε οι χρήστες θα πρέπει να μπορούν να βρουν αξεσουάρ ανά τύπο, όνομα ή τοποθεσία στο σπίτι.

3.6 Ενσωμάτωση Siri

Η Siri μπορεί να κάνει εύκολη την εκτέλεση περίπλοκων εργασιών με μια μόνο πρόταση. Η Siri αναγνωρίζει ονόματα Σπιτιών, Δωματίων και Ζωνών και μπορεί να υποστηρίξει προτάσεις όπως “Siri, lock up my house in Tahoe”, “Siri, turn off the upstairs lights”, και “Siri, make it warmer in the media room”. Επίσης η Siri αναγνωρίζει ονόματα αξεσουάρ και χαρακτηριστικών, έτσι ώστε οι χρήστες να μπορούν να δώσουν εντολές όπως “Siri, dim the desk lamp”.

Είναι καλό να γνωρίζουν οι χρήστες ποιες ενέργειες μπορούν να εκτελεστούν από την Siri κατά την διάρκεια διαμόρφωσης της ενέργειας.

[Πηγή: 5]

Κεφάλαιο 4

Εφαρμογή Arduino

4.1 Περιγραφή aJSON & JSON

Το aJSON είναι μια προσπάθεια να μεταφερθεί μια πλήρης εκτέλεση του JSON στο Arduino. Βασίζεται στο cJson με την διαφορά ότι είναι μικρότερο σε μέγεθος και με μια-δυο δυνατότητες λιγότερες :

- Το μέγιστο μέγεθος ενός πίνακα ή μιας λίστας μπορεί να αποτελείται από 255 στοιχεία
- Υπάρχει ένας εσωτερικός buffer ο οποίος δεσμεύει μέχρι και 256 bytes από την ram

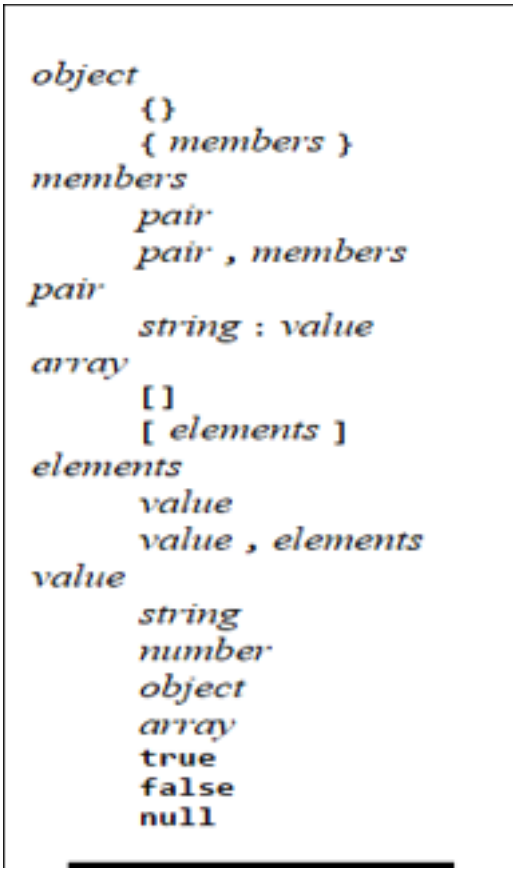
Το JSON(JavaScript Object Notation) είναι μια απλή μορφή ανταλλαγής δεδομένων. Είναι εύκολο για τους ανθρώπους να το διαβάσουν και να γράψουν σε αυτό. Είναι επίσης εύκολο για τους υπολογιστές να το αναλύσουν και να το παράγουν. Βασίζεται σε ένα υποσύνολο της JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. Το JSON είναι μια μορφή κειμένου που είναι πλήρως ανεξάρτητη από τις γλώσσες προγραμματισμού, αλλά χρησιμοποιεί στοιχεία που είναι οικεία στους προγραμματιστές της οικογένεια γλώσσας C, στην οποία συμπεριλαμβάνονται οι C, C++, C#, Java, Javascript, Perl, Python και άλλες πολλές. Αυτές οι ιδιότητες κάνουν το JSON μια ιδανική γλώσσα για ανταλλαγή δεδομένων.

Το JSON αποτελείται από δύο δομές :

- Μια συλλογή από ονόματα/τιμές-αξίες. Σε διάφορες γλώσσες, αυτό αναγνωρίζεται ως ένα αντικείμενο, μια εγγραφή, μια δομή, ένα λεξικό, ένα hash table, ένας πίνακας συσχετισμών.
- Μια διατεταγμένη λίστα με τιμές. Στις περισσότερες γλώσσες αυτό αναγνωρίζεται ως ένας πίνακας, διάνυσμα, λίστα ή ακολουθία.

Αυτές είναι δομές που είναι εφαρμόσιμες σε όλες τις γλώσσες. Θεωρητικά όλες οι σύγχρονες γλώσσες προγραμματισμού υποστηρίζουν αυτές τις δομές με τον έναν ή

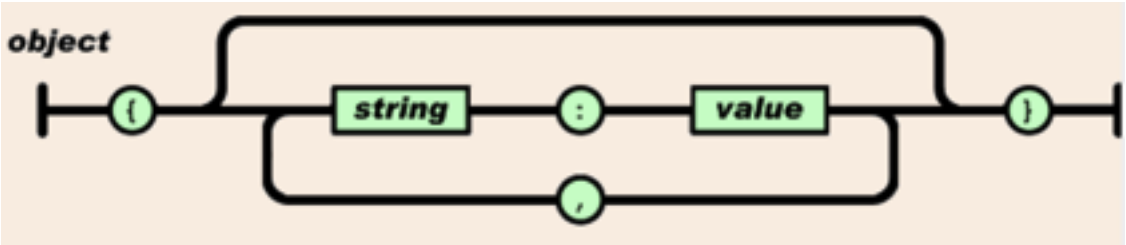
τον άλλον τρόπο. Είναι λογικό μια μορφή δεδομένων που μπορούν να την ανταλλάξουν οι γλώσσες προγραμματισμού μεταξύ τους να είναι βασισμένη σε αυτές τις δομές.



Εικόνα 4.1 : Διάφορες μορφές δομών

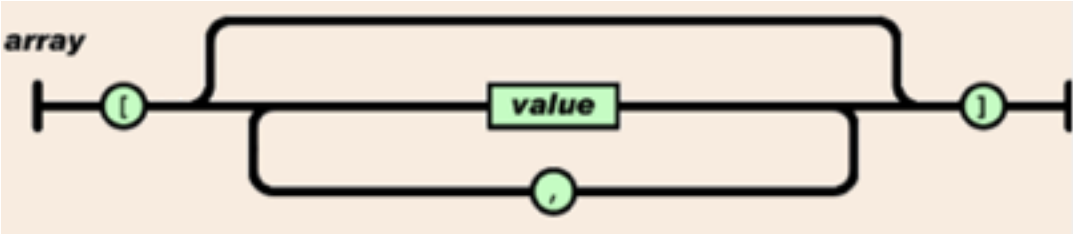
Στο JSON υπάρχουν οι εξής μορφές :

- Ένα αντικείμενο το οποίο είναι μια μη διατεταγμένη συλλογή από ονόματα/τιμές-αξίες. Ένα αντικείμενο ξεκινάει με άγκιστρο “{“ και τελειώνει με άγκιστρο “}”. Στο τέλος κάθε ονόματος μπαίνει άνω και κάτω τελεία “:” και το διαχωριστικό σύμβολο μεταξύ ονόματος και τιμής είναι το κόμμα “,”.



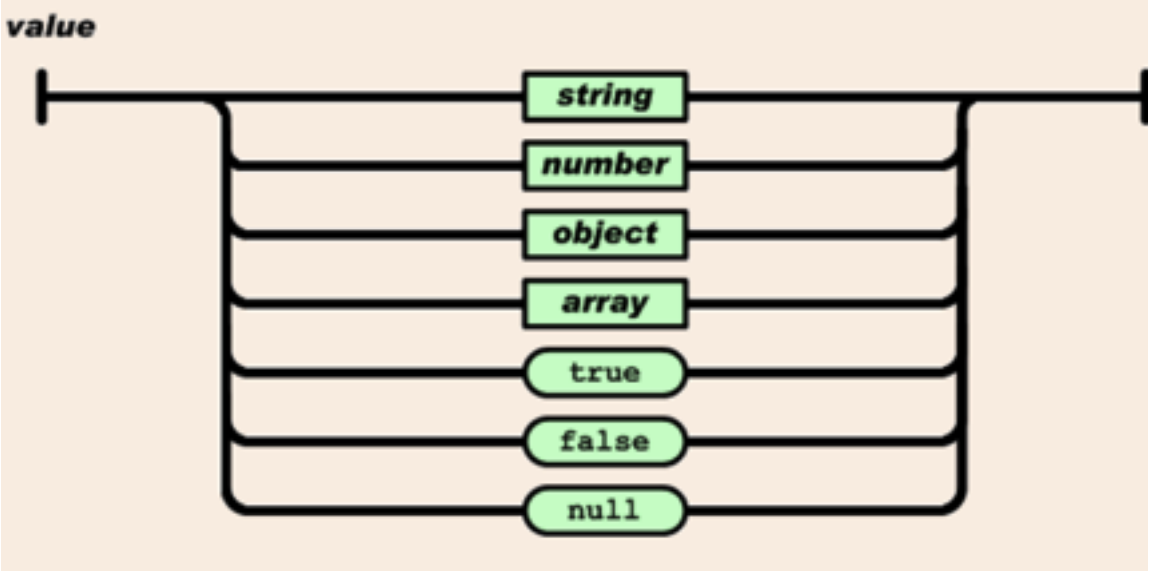
Εικόνα 4.2 : Διάγραμμα δομής ενός αντικειμένου

- Ένας πίνακας είναι μια διατεταγμένη συλλογή από τιμές, ο οποίος αρχίζει με αγκύλη “[“ και τελειώνει με αγκύλη “]”. Οι τιμές διαχωρίζονται με κόμμα “,”.



Εικόνα 4.3 : Διάγραμμα δομής ενός πίνακα

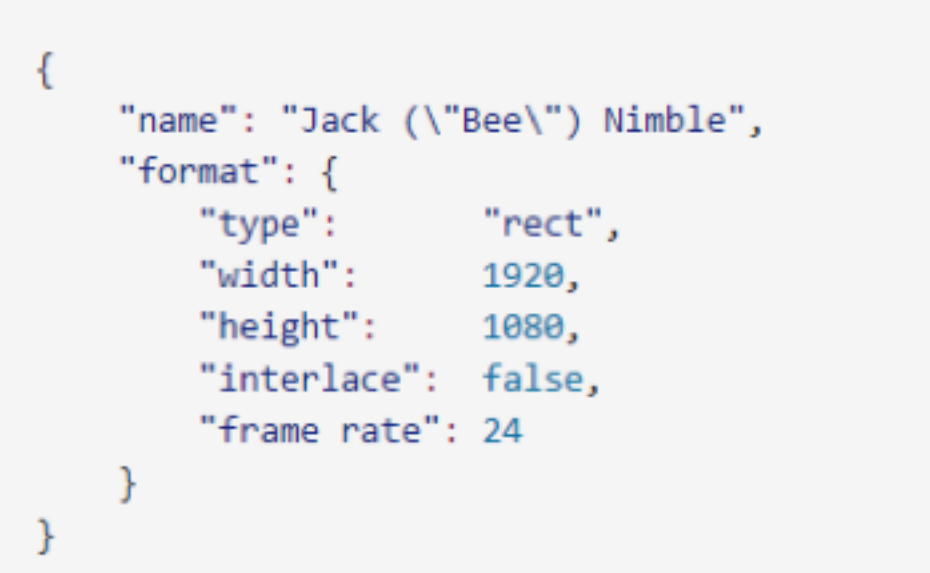
- Μια τιμή μπορεί να είναι ένα αλφαριθμητικό, ή αριθμός, ή true/false/null, ή αντικείμενο ή πίνακας.



Εικόνα 4.4 : Τύποι τιμών

Το JSON είναι κάτι σαν την XML. Χρησιμοποιείται για να μεταφέρεις δεδομένα, να αποθηκεύσεις δεδομένα, ή απλά για να παρουσιάσεις την κατάσταση του προγράμματος σου. Το JSON είναι χρήσιμο, ειδικά για να ανταλλάξεις δεδομένα αποτελεσματικά σε γλώσσες προγραμματισμού όπως τις JavaScript,Java, C++.
Το aJson είναι μια βιβλιοθήκη για να δεχθείς, να αναγνωρίσεις, να δημιουργήσεις αλλά και να επεξεργαστείς αλφαριθμητικά κατευθείαν μέσα στο Arduino.

Το aJson προσφέρει την δυνατότητα να αναλύσεις τα αλφαριθμητικά του JSON σε αντικείμενα.



Εικόνα 4.5 : Αναπαράσταση της μορφής JSON

[Πηγή: 6, Πηγή: 7]

4.2 Ανάλυση του JSON

Για να αναλύσεις μια δομή JSON μέσω του aJson, απλά μετατρέπεις την δομή σε μια σειρά(δέντρο) αντικειμένων :

aJsonObject* jsonObject = aJson.parse(json_string);

Υποθέτουμε ότι η δομή JSON που θέλουμε να αναλύσουμε βρίσκεται στην μεταβλητή json_string ως χαρακτήρας.

Αυτό είναι ένα αντικείμενο, στην C δεν υπάρχουν αντικείμενα, υπάρχουν όμως δομές. Έτσι τα αντικείμενα μεταφράζονται-μετατρέπονται σε δομές, μαζί με τα μειονεκτήματα της.

Για παράδειγμα τώρα μπορούμε να ανακτήσουμε την τιμή του ονόματος :

aJsonObject* name = aJson.getObjectItem(root,”name”);

Η τιμή του ονόματος μπορεί να ανακτηθεί μέσω :

Serial.println(name->valuestring);

Σημειώστε ότι το aJsonObject έχει οργανωμένα όλους τους τύπους των τιμών, έτσι ώστε να μπορείτε να πάρετε μόνο τα χρήσιμα δεδομένα από τον τύπο τιμών που θέλετε:

name->type
το οποίο μπορεί είναι aJson_False, aJson_True, aJson_NULL, aJson_Number, aJson_String, aJson_Array ή aJson_Object.

Για το aJson_Number μπορεί να χρησιμοποιηθεί το value.number.valueint ή value.number.valuedouble
Για το aJson_String μπορεί να χρησιμοποιηθεί το value.valuestring
Για το True και False μπορεί να χρησιμοποιηθεί το value.valuebool
Για να κάνετε το αντικείμενο String απλά καλείται η εξής συνάρτηση :
Char *json_String = aJson.print(jsonObject);

Καθώς οι επεξεργαστές του arduino έχουν πολύ μικρές χωρητικότητες μνήμης ram και τα struct που δημιουργούνται μετά από κάθε ανάλυση ενός αλφαριθμητικού JSON καταλαμβάνουν μεγάλο όγκο είναι αναγκαίο να διαγράφεται η ρίζα του αντικειμένου μετά από κάθε μετατροπή ενός αντικειμένου σε δομή:
aJson.deleteItem(root);

4.2.1 Ανάλυση ροής

Αποθηκεύοντας το String με το κανονικό του μέγεθος και το αντικείμενο JSON, θα δεσμευτεί πολύ μνήμη στο και αυτό δεν συμφέρει επειδή τα Arduino δεν έχουν μεγάλη μνήμη ram. Επομένως είναι καλύτερο να αναλύσουμε την ροή των δεδομένων αντί να αναλύσουμε τα Strings. Η ροή στην C είναι όπως το FILE που είναι ένας ειδικός τύπος δεδομένων το οποίο δίνει την δυνατότητα πρόσβασης σε αρχεία. Για παράδειγμα αν χρησιμοποιηθεί το FILE :
aJsonObject* jsonObject = aJson.parse(file);
Με αυτό τον τρόπο δεν αποθηκεύεται το JSON String στην μνήμη.

4.2.2 Φιλτράρισμα κατά την διάρκεια της ανάλυσης

Κάθε JSON μπορεί να έχει μια σειρά από ονόματα/τιμές τα οποία ένας κώδικας μπορεί να μην τα χρειάζεται είτε να μην τα καταλαβαίνει. Για να αποφευχθεί η αποθήκευση αυτών των τιμών στην μνήμη, χρησιμοποιούνται φίλτρα κατά την διάρκεια της ανάλυσης. Ένα φίλτρο μπορεί να αποτελείται από ένα σύνολο ονομάτων τα οποία χρειάζεσαι. Εάν για παράδειγμα ενδιαφέρεσαι για τα “name”, “format”, “height”, “width” τα οποία θα τελειώνουν με NULL:

Char** jsonFilter = {“name,”format”,”height”,”width”,NULL};

aJsonObject* jsonObject = aJson.parse(json_string,json_Filter);

ας υποθέσουμε ότι αποθηκεύσαμε το JSON String στην μεταβλητή json_string ως χαρακτήρα

Με αυτό τον τρόπο μόνο η παρακάτω δομή θα αναλυθεί, τα άλλα δεδομένα θα αγνοηθούν:

```
{
    "name": "Jack (\"Bee\") Nimble",
    "format": {
        "width":      1920,
        "height":     1080,
    }
}
```

Εικόνα 4.6 : Παράδειγμα δομής ενός αντικειμένου JSON

4.2.3 Δημιουργία αντικειμένων JSON

```
aJsonObject *root,*fmt;
root=aJson.createObject();
aJson.addItemToObject(root, "name", aJson.createItem("Jack (\"Bee\") Nimble"));
aJson.addItemToObject(root, "format", fmt = aJson.createObject());
aJson.addStringToObject(fmt,"type",     "rect");
aJson.addNumberToObject(fmt,"width",        1920);
aJson.addNumberToObject(fmt,"height",       1080);
aJson.addFalseToObject (fmt,"interlace");
aJson.addNumberToObject(fmt,"frame rate",   24);
```

Εικόνα 4.7 : Αναπαράσταση δημιουργίας ενός αντικειμένου JSON

Η ρίζα του αντικειμένου : είναι τύπου Object και έχει ένα “παιδί” το οποίο ονομάζεται “name”, με τιμή “Jack(“Bee”) Nimble”, και ένα δεύτερο “παιδι” που είναι τύπου Object και ονομάζεται “format” το οποίο έχει τα εξής παιδιά :

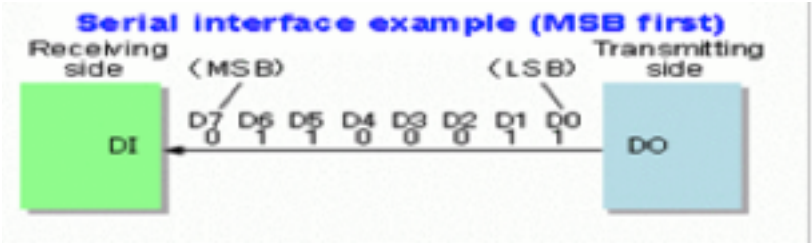
```
aJsonObject* root = aJson.createArray();

aJsonObject* day;
day=aJson.createItem("Monday");
aJson.addItemToArray(root, day);
day=aJson.createItem("Tuesday");
aJson.addItemToArray(root, day);
day=aJson.createItem("Wednesday");
aJson.addItemToArray(root, day);
day=aJson.createItem("Thursday");
aJson.addItemToArray(root, day);
day=aJson.createItem("Friday");
aJson.addItemToArray(root, day);
day=aJson.createItem("Saturday");
aJson.addItemToArray(root, day);
day=aJson.createItem("Sunday");
aJson.addItemToArray(root, day);
```

Εικόνα 4.8 : Αναπαράσταση δημιουργίας πίνακα JSON

Το πρώτο παιδί είναι τύπου String και ονομάζεται type και η τιμή του είναι το “rect”, το άλλο παιδί είναι τύπου Number, ονομάζεται width και η τιμή του είναι το “1920”, το άλλο παιδί είναι τύπου Number, ονομάζεται height και η τιμή του είναι “1080”, το άλλο παιδί είναι τύπου False, ονομάζεται interlace και το τελευταίο παιδί είναι τύπου Number, ονομάζεται Frame rate και έχει τιμή “24”.

Η δημιουργία πίνακα έχει την ίδια λογική :

Το λεγόμενο Filter Parsing δηλαδή, αυτό που κάνει είναι να δέχεται ένα αλφαριθμητικό με συγκεκριμένη δομή(τύπου δέντρου). Η συγκεκριμένη δομή διευκολύνει την ανάλυση του αλφαριθμητικού και συνεπώς και την απομόνωση των



δεδομένων που χρειάζονται βάσει του φίλτρου που έχει χρησιμοποιηθεί. Το αποτέλεσμα της ανάλυσης αυτής είναι η κατανόηση του αλφαριθμητικού (το οποίο στην ουσία έχει μεταφραστεί από την γλώσσα εισόδου, στην επιθυμητή γλώσσα) για να εκτελεστούν οι ενέργειες που θέλουμε.

[Πηγή: 6]

4.3 Εγκατάσταση και συνδεσμολογία

Σειριακή επικοινωνία είναι η διαδικασία αποστολής δεδομένων του ενός bit τη φορά, μέσω ενός καναλιού επικοινωνίας.

Εικόνα 4.9 : Αναπαράσταση σειριακής επικοινωνίας

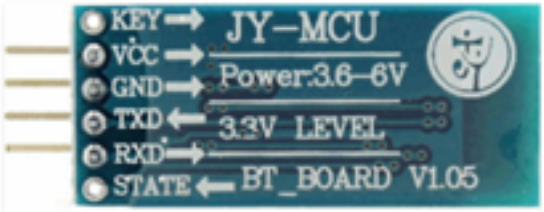
Bluetooth είναι μια τεχνολογία ασύρματης επικοινωνίας (ανταλλαγής δεδομένων) κοντινών αποστάσεων (χρησιμοποιώντας μικρού μήκους κύματος UHF ραδιοκύματα στο εύρος ISM που ξεκινάνε από τα 2.4 GHz και φτάνουν μέχρι 2.485 GHz), δημιουργώντας έτσι δίκτυα μικρών διαστάσεων με εμβέλεια περίπου 10 μέτρων

[Πηγή: 8]

4.3.1 Συνδεσμολογία του Bluetooth με το Arduino ATmega644P

VCC : Τροφοδοσία του εξαρτήματος Bluetooth. Την τροφοδοσία την παρέχει το Arduino δίνοντας 5V.

GND: γείωση. Την γείωση την παρέχει το Arduino.



TXD : Αυτό το pin χρησιμοποιείται για να στείλει δεδομένα το Bluetooth στο Arduino. Αυτό το pin συνδέεται στο pin RX του Arduino, απο το οποίο μπορεί να δέχεται δεδομένα σειριακά το Arduino.
RXD : Αυτό το pin χρησιμοποιείται για να δέχεται δεδομένα το Bluetooth από το Arduino. Συνδέεται στο pin TX του Arduino.

Εικόνα 4.10 : Pins του Bluetooth module HC-06

Στην παραπάνω εικόνα φαίνονται τα Pins του Bluetooth εξαρτήματος. Βλέπουμε ότι το TXD λειτουργεί με 3.3V, αυτό σημαίνει ότι παρόλο που εμείς θα το τροφοδοτήσουμε με 5V, εκείνο θα λειτουργεί με 3.3V.

4.3.2 Ρυθμίσεις επικοινωνίας

1. Ανοίγουμε την διεπαφή Arduino και πάμε στο monitor, αφού έχουμε συνδεδεμένο μόνο το Bluetooth .
2. Θα πρέπει να δούμε το εξής : Type AT commands. Αν δεν το δούμε αυτό υπάρχει κάποιο πρόβλημα στην συνδεσμολογία.
3. Πληκτρολογούμε AT στο serial monitor και πατάμε enter. Θα πρέπει να απαντήσει OK.
4. Αλλάζουμε το baud rate επικοινωνίας που θα έχει το Bluetooth me to Arduino, πληκτρολογώντας AT+BAUD7 (57600), πληκτρολογούμε πάλι AT, για να επιβεβαιώσουμε την αλλαγή. Επίσης επιβεβαιώνουμε ότι το baud rate επικοινωνίας του υπολογιστή με το Bluetooth είναι 9600.

Το baud rate είναι η ταχύτητα επικοινωνίας (ανταλλαγής δεδομένων) μεταξύ του Arduino και του Bluetooth.

4.3.3 Σύνδεση με υπολογιστή

1. Αφού έχουμε συνδέσει το Bluetooth με το ATmega644P, τροφοδοτούμε το Arduino, συνεπώς και το Bluetooth και αρχίζει να αναβοσβήνει το led που βρίσκεται πάνω στο Bluetooth ως ένδειξη ότι τροφοδοτήθηκε.
2. Συνδέουμε το Bluetooth με το laptop. Αντικαθιστούμε την προεπιλεγμένη ονομασία του Bluetooth με το αξεσουάρ που το έχουμε αντιστοιχήσει και αλλάζουμε

και τον προεπιλεγμένο κωδικό του Bluetooth που είναι το 1234, για λόγους ασφαλείας.

3. Αφού ο υπολογιστής και το Bluetooth έχουν συνδεθεί, στο τερματικό θα πρέπει να δούμε ένα μήνυμα το οποίο λέει πως συνδέθηκαν επιτυχώς και τότε το led του Bluetooth που αναβόσβηνε, θα μείνει σταθερά αναμμένο ως ένδειξη επιτυχημένης σύνδεσης με τον υπολογιστή

4.4 Επεξήγηση του κώδικα

Ξεκινάμε με την δήλωση των μεταβλητών του ρελέ και του touch sensor.

Στην συνάρτηση setup: ορίζουμε το baud rate που θα έχει το Bluetooth με τον ATmega644P, που είναι 57600 στο serial1. Επίσης ορίζουμε το baud rate του Bluetooth με τον υπολογιστή στο serial, το οποίο χρησιμοποιούμε για την εμφάνιση των πιθανών σφαλμάτων.

Αρχικοποιούμε τα pins του ρελέ(ως έξοδος) και του touch sensor(ως είσοδος).

Στην συνέχεια φτιάχνουμε μια δομή με τα αξεσουάρ που θα χρησιμοποιήσουμε, τα οποία αριθμούμε για να εκτελέσουμε διαφορετικές ενέργειες ανάλογα το αξεσουάρ. Για το κάθε αξεσουάρ θα υπάρχει διαφορετική αντιμετώπιση, η ανάλυση όμως ακολουθεί την ίδια λογική. Δηλαδή αφού αποθηκεύσουμε σε μια μεταβλητή το αλφαριθμητικό το οποίο θέλουμε να αναλύσουμε, αρχίζουμε να το χωρίζουμε σε κομμάτια. Το αλφαριθμητικό που θα λάβουμε είναι σε μορφή αντικειμένων (object) την οποία θα την μετατρέψουμε σε μορφή δομής δεδομένων (struct).

- Ξεκινάμε από την ρίζα του αντικειμένου (root).
- Αναγνωρίζουμε το όνομα του αντικειμένου, όπου ανάλογα το όνομα θα προχωρήσουμε σε διαφορετικό κομμάτι κώδικα, αφού το όνομα δηλώνει και το αξεσουάρ.
- Προχωράμε στην ανάλυση των ιδιοτήτων-πληροφοριών του αξεσουάρ, όπως την κατάσταση του, η φωτεινότητα.
- Ανάλογα τις τιμές αυτών των πληροφοριών, εκτελούμε κάποιες ενέργειες.

Εάν για παράδειγμα το αλφαριθμητικό που θα δεχτούμε είναι για το αξεσουάρ της λάμπας και τιμή της κατάστασης του είναι ON, τότε θα ανάψει η λάμπα και θα επιστρέψουμε το μήνυμα “ok” στον υπολογιστή μέσω του Bluetooth, ως ένδειξη ότι

ολοκληρώθηκε με επιτυχία το αίτημα που στάλθηκε και τέλος καταστρέφουμε το αλφαριθμητικό διαγράφοντας την ρίζα του αντικειμένου.
Σε περίπτωση που δεν δημιουργηθεί η ρίζα του αντικειμένου από το αλφαριθμητικό που θα σταλεί, τότε στέλνουμε το μήνυμα “fail” στον υπολογιστή.

Void loop()

Οι μηχανικοί διακόπτες όπως και οι διακόπτες αφής, μερικές φορές παράγουν μεταβάσεις από low σε high(πάτημα διακόπτη) λανθασμένα. Αυτό που συμβαίνει ακριβώς είναι ότι ο διακόπτης μπορεί να αντιληφθεί το ένα πάτημα ως πολλαπλά και οφείλεται σε φυσικά αίτια.

Η λύση του προβλήματος είναι να ελέγχουμε την κατάσταση του διακόπτη μετά από ένα πάτημα, για χρονικό διάστημα μερικών millisecond, έτσι ώστε να εξασφαλίσουμε ότι η μετάβαση της κατάστασης οφείλεται σε πάτημα του διακόπτη και όχι σε στατικό ηλεκτρισμό.

Κάθε φορά που η είσοδος του διακόπτη αφής που χρησιμοποιούμε αλλάζει τιμή από low σε high όταν το πατάμε, η έξοδος του αλλάζει και αυτή από low σε high ή το αντίστροφο.

Φτιάχνουμε δύο μεταβλητές, στις οποίες θα αποθηκεύσουμε την τρέχουσα και την προηγούμενη κατάσταση του διακόπτη (την οποία αρχικοποιούμε με λογικό 0). Επίσης φτιάχνουμε άλλες δύο μεταβλητές στις οποίες θα αποθηκεύσουμε τον χρόνο που άλλαξε τελευταία φορά τιμή η έξοδος του διακόπτη
Αναλυτικότερα στον κώδικα αυτό που συμβαίνει είναι το εξής :

Δημιουργούμε δύο μεταβλητές, όπου η πρώτη θα αποθηκεύει τον χρόνο που άλλαξε τελευταία φορά κατάσταση ο διακόπτης και η δεύτερη έχει το όριο των 50 millisecond, που είναι και το όριο ασφαλείας αλλαγής κατάστασης του διακόπτη. Καταχωρούμε σε μεταβλητές την είσοδο της προηγούμενης κατάστασης του διακόπτη, αλλά και την τρέχουσα. Ελέγχουμε την κατάσταση εισόδου του διακόπτη και την συγκρίνουμε με την προηγούμενη κατάσταση της και καταχωρούμε τον χρόνο(την ώρα) που άλλαξε κατάσταση ο διακόπτης. Στη συνέχεια ελέγχουμε αν η μετάβαση της κατάστασης έγινε σε χρονικό διάστημα μεγαλύτερο των 50 millisecond, συγκρίνοντας τον τρέχον χρόνο, με τον χρόνο που έγινε η αλλαγή της κατάστασης. Εάν η αλλαγή της κατάστασης έγινε σε περισσότερο από 50 millisecond, τότε η αλλαγή οφείλεται σε πάτημα του διακόπτη και όχι σε θόρυβο. Αποθηκεύουμε την

τρέχουσα κατάσταση του διακόπτη στην μεταβλητή που έχουμε ορίσει και εάν αυτή η κατάσταση έχει λογικό “1” τότε αντιστρέφουμε την τελευταία κατάσταση του ρελέ, εάν η τρέχουσα κατάσταση του ρελέ έχει λογικό “1” τότε ενημερώνουμε την εφαρμογή για την κατάσταση του ρελέ, στέλνοντας ένα αλφαριθμητικό όπως αυτό που λάβαμε και αναλύσαμε.

Στην περίπτωση που ο χρόνος μετάβασης της κατάστασης του διακόπτη δεν ξεπερνά τα 50 millisecond, σημαίνει ότι η μετάβαση οφείλεται σε θόρυβο, καταχωρούμε την τρέχουσα κατάσταση του διακόπτη στην μεταβλητή της προηγούμενης κατάστασης διακόπτη ως σημείο ελέγχου της επόμενης αλλαγής του διακόπτη.

Παρακάτω ελέγχουμε αν έχουν σταλεί στο Bluetooth δεδομένα που πρέπει να διαβάσουμε. Τα αποθηκεύουμε σε μια μεταβλητή και τα μετατρέπουμε σε αλφαριθμητικό από χαρακτήρες. Στην συνέχεια καλούμε την συνάρτηση parseJson, με όρισμα το αλφαριθμητικό που μετατρέψαμε. Η συνάρτηση επιστρέφει “ok” ή “fail” ανάλογα αν πέτυχε ή αν απέτυχε η ανάλυση του αλφαριθμητικού.

4.4.1 aJSON reverse engineer

Ξεκινάμε ψάχνοντας την συνάρτηση parse στο αρχείο aJSON.h, το οποίο και ενσωματώνουμε στον κώδικα μας. Βρίσκουμε ότι η συνάρτηση parse, βρίσκεται στην κλάση aJsonClass, στο τμήμα public που δηλώνει ότι είναι καθολικά ορατό. Η συνάρτηση parse είναι τύπου aJsonObject*, δηλαδή επιστρέφει δείκτη σε ένα aJson αντικείμενο και παίρνει σαν όρισμα ένα αλφαριθμητικό (string).

Πηγαίνουμε στο αρχείο .cpp να βρούμε την υλοποίηση της συνάρτησης parse. Βλέπουμε το aJsonClass:parse(char *value), που σημαίνει ότι έχουμε πρόσβαση στην υλοποίηση της συνάρτησης Parse.

Στην υλοποίηση υπάρχει μια συνάρτηση τύπου aJsonStringStream με όνομα stringStream που παίρνει δύο ορίσματα(αλφαριθμητικό,NULL)

Επίσης υπάρχει άλλη μια συνάρτηση parse που παίρνει σαν όρισμα την διεύθυνση του αλφαριθμητικού και επιστρέφει μια νέα μεταβλητή η οποία είναι τύπου aJsonObject και περιέχει το parent JSON object του string που δώσαμε να αναλυθεί.

Βρίσκουμε την κλάση aJsonStringStream στο αρχείο .h ,στην οποία είναι εμφωλευμένη άλλη μια κλάση που λέγεται aJsonStream, η οποία βρίσκεται στο

αρχείο .cpp .Όπου υλοποιείται η συνάρτηση parsevalue η οποία καλεί και άλλες συναρτήσεις έτσι ώστε να αναλυθεί το αντικείμενο το οποίο διαβάζουμε.

Η υλοποίηση έχει ως εξής :

Ξεκινάει να διαβάζει έναν-έναν τους χαρακτήρες ως ακεραίους αριθμούς. Για να αναλυθεί το περιεχόμενο, υπάρχουν κάποιες συνθήκες.

- Σε περίπτωση που διαβαστεί ο τερματικός χαρακτήρας, σημαίνει ότι τελείωσε η ανάγνωση και επιστρέφεται το αλφαριθμητικό.
- Όταν διαβαστεί αριθμός, θα εξεταστεί αν είναι ακέραιος ή δεκαδικός, το πρόσημο του αριθμού και θα επιστραφεί η τιμή του.
- Εάν διαβαστεί το “{“ ή “}”, σημαίνει ότι δημιουργήθηκε ένα νέο παιδί αντικείμενο. Με το “,” ξεχωρίζει ένα άλλο παιδί αντικείμενο του ίδιου γονέα. Αν μετά από κάποιο αντικείμενο ακολουθεί το “:”, σημαίνει ότι μετά από αυτό ακολουθεί η τιμή του αντικειμένου και το “}” σηματοδοτεί το τέλος της ανάγνωσης παιδιών αντικειμένων
- Εάν διαβαστεί το “[“ ή “]”. Σημαίνει ότι κάποιο αντικείμενο έχει τιμή-αξία έναν πίνακα, με πολλαπλές τιμές οι οποίες χωρίζονται μεταξύ τους με κόμμα “,”.
- Επίσης μπορεί να διαβαστούν Boolean τιμές ενός αντικειμένου όπως true ή false. Ακόμη η τιμή ενός αντικειμένου μπορεί να είναι NULL. Και σε αυτή την περίπτωση επιστρέφουμε το περιεχόμενο(τιμή) του αντικειμένου.

Κεφάλαιο 5

Εφαρμογή NodeJS

Το σύστημα επικοινωνίας που έχουμε δημιουργήσει απαιτεί την δημιουργία ενός αξεσουάρ το οποίο θα είναι συμβατό με τον τρόπο και τα πρωτόκολλα λειτουργίας του Homekit®, για αυτό τον σκοπό θα χρησιμοποιήσουμε ένα ειδικά σχεδιασμένο http server στον οποίο έχουν προστεθεί όλες οι διαδικασίες που χρησιμοποιεί το Homekit® για την ταυτοποίηση ενός αξεσουάρ. Το όνομα αυτού του server είναι HAP-NodeJS και είναι διαθέσιμο ως λογισμικό ανοιχτού κώδικα στο GitHub®. Ο λόγος που χρησιμοποιήθηκε αυτό το λογισμικό είναι προφανής αν σκεφτούμε πως η δημιουργία ενός αξεσουάρ Homekit® μέσω επισήμων οδών(βλέπε: Apple MFI®) θα κόστιζε μερικές χιλιάδες ευρώ. Αυτή η εφαρμογή μας δίνει την δυνατότητα να δημιουργούμε εικονικά αξεσουάρ μέσω του NodeJS τα οποία είναι “ορατά” στον μηχανισμό του Homekit® σε Apple συσκευές, τα οποία με την σειρά τους στέλνουν εντολές στις πραγματικές συσκευές μετά από κάθε ενέργεια του χρήστη, δίνοντας μας την δυνατότητα να δημιουργήσουμε αξεσουάρ με πρόσβαση στο πρωτόκολλο Homekit®, χωρίς την ύπαρξη επίσημης ταυτοποίησης από την Apple. Αυτός ο http server έχει την δυνατότητα να τρέξει σε κάθε λειτουργικό που υποστηρίζει η πλατφόρμα NodeJS, κάνοντας το εξαιρετικά ισχυρό και ικανό να τρέξει από συσκευές όπως το Raspberry Pi, καθιστώντας το έναν εξαιρετικό τρόπο υλοποίησης ακόμα και ενός εμπορικού προϊόντος, μιας και μπορεί να τρέξει από μια εξαιρετικά μικρή συσκευή. Οι βασικές κλάσεις που χρησιμοποιούμε από τον server αυτό είναι οι: Accessory, Bridge, Service, Characteristic. Αυτές οι κλάσεις εμπεριέχουν μεθόδους που επιτρέπουν την δημιουργία εξαρτημάτων και group εξαρτημάτων που ονομάζονται και γέφυρες και της προσθήκης υπηρεσιών και χαρακτηριστικών σε αυτά τα εξαρτήματα.

Δυστυχώς παρόλο που ο server αυτός δουλεύει πολύ καλά για την δημιουργία αξεσουάρ που ελέγχονται από το τοπικό WiFi δίκτυο, δεν έχει την δυνατότητα να συνδεθεί με τον ασύρματο μηχανισμό ελέγχου της Apple, που ονομάζεται Homekit over iCloud. Είναι όμως εφικτή η χρήση αυτού του μηχανισμού με την προϋπόθεση ύπαρξης ενός Apple TV (3ης γενιάς και άνω), ο οποίος δρα ως ενδιάμεσος του iCloud με το HAP-NodeJS και επιτρέπει και τον απομακρυσμένο έλεγχο των συσκευών μας με το να τα κάνει ορατά στο iCloud το Apple TV.

Χρησιμοποιούμε λοιπόν, τον http server για την επικοινωνία με το Homekit, είτε αυτό είναι μέσω του τοπικού δικτύου είτε είναι μέσω του iCloud και χρησιμοποιούμε το πρωτόκολλο Bluetooth® για την επικοινωνία του υπολογιστή πάνω στον οποίο λειτουργεί ο http server με τα πραγματικά μας αξεσουάρ. Για την επικοινωνία του υπολογιστή με το bluetooth των εξαρτημάτων μας χρησιμοποιούμε την βιβλιοθήκη του nodeJS, που ονομάζεται “bluetooth-serial-port” και χρησιμοποιεί την δυνατότητα του πρωτοκόλλου bluetooth για την δημιουργία επικοινωνίας τύπου κονσόλας, μεταξύ δύο bluetooth module(του υπολογιστή και των εξαρτημάτων μας) για την αποστολή εντολών σε plain-text, που υπαγορεύουν στο εξάρτημα τι ενέργειες θα εκτελέσει. Οι εντολές αυτές αποστέλλονται σε μορφή λεκτικών που αντιπροσωπεύουν αντικείμενα JSON, αντικείμενα τα οποία είναι εύκολα προσβάσιμα σε εμάς μιας και γράφουμε σε JavaScript αλλά και εύκολα μεταφραζομένων από την γλώσσα C η οποία τρέχει στο εξάρτημα τύπου Arduino.

5.1 Τεχνική ανάλυση βασικών κλάσεων

Φάκελος Lib:

Characteristic.js: Αυτή η κλάση είναι υπεύθυνη για την δημιουργία των χαρακτηριστικών ενός αξεσουάρ. Χρησιμοποιείται για την δημιουργία αντικειμένων τα οποία είναι υπεύθυνα για την διαχείριση ενός συγκεκριμένου χαρακτηριστικού. Αυτά τα αντικείμενα περιέχουν μεθόδους που επιτρέπουν την ανάγνωση των τιμών οποιουδήποτε χαρακτηριστικού και την αποστολή του στις συνδεδεμένες Homekit® συσκευές αλλά και την ενημέρωση της νέας τιμής ενός χαρακτηριστικού σε περίπτωση αλλαγής του. Ακόμα περιέχει μεθόδους οι οποίες καλούνται μετά από κάποια ενέργεια της iOS συσκευής, για την αλλαγή της τιμής του συγκεκριμένου χαρακτηριστικού. Τα αντικείμενα αυτής της κλάσης είναι απαραίτητο να αποτελούν μέρος ενός αντικειμένου Service, προκειμένου να μπορούν να είναι προσβάσιμα από την iOS συσκευή. Περιέχει μια μέθοδο, την toHAP προκειμένου να φτιάξει ένα λεκτικό το οποίο επιτρέπει στο Homekit να αναγνωρίσει το χαρακτηριστικό βάση του πρωτοκόλλου που υλοποιεί. Τέλος σε κάθε αλλαγή της τιμής ενός χαρακτηριστικού και σε κάθε ενημέρωση της τιμής του, φροντίζει να ενημερώνει με τα event get,set,change κάθε άλλη κλάση που “ακούει” αυτού του είδους τα event.

Service.js: Αυτή η κλάση είναι υπεύθυνη για την δημιουργία Services, τα οποία μπορούν να προστεθούν σε αξεσουάρ προκειμένου να φανερώσουν τα χαρακτηριστικά ενός αξεσουάρ στο Homekit. Είναι ουσιαστικά μια διευκόλυνση που επιτρέπει την ομαδοποίηση πολλών χαρακτηριστικών, για να κάνει πιο εύκολο τον έλεγχο τους. Η κλάση αυτή περιέχει μεθόδους που επιτρέπουν την προσθαφαίρεση χαρακτηριστικών και την εξέταση ύπαρξης τους μέσα σε μια υπηρεσία. Τέλος επιτρέπουν την εύρεση και την επιστροφή ενός αντικειμένου χαρακτηριστικού από την στοίβα χαρακτηριστικών, που διατηρεί προκειμένου να χειριστούμε το χαρακτηριστικό εκείνο.

Accessory.js: Αυτή η κλάση είναι από τις πιο σημαντικές της συγκεκριμένης υλοποίησης του http server. Αυτή η κλάση είναι η μοναδική που αποκαλύπτεται μέσω του http server στο Homekit και αναλαμβάνει να εμφανίζει τις υπηρεσίες και τα χαρακτηριστικά που περιλαμβάνει στο Homekit. Περιέχει το Service που είναι υποχρεωτικό να έχουν όλα τα αξεσουάρ, το AccessoryInformation, το οποίο εμπεριέχει το όνομα, τον κατασκευαστή, το μοντέλο και τον σειριακό αριθμό του αξεσουάρ. Ενώ διαθέτει μεθόδους για την προσθαφαίρεση περισσοτέρων υπηρεσιών, την προσθήκη αξεσουάρ σε διαμόρφωση “γέφυρας”, η κατάσταση όπου ένα αξεσουάρ ελέγχει πολλαπλά άλλα, ενώ διαθέτει και πολλαπλές μεθόδους οι οποίες αντιδρούν σε μηνύματα από τον http server, προκειμένου να αναγνωστούν τα μηνύματα που αποστέλλονται σε κάθε accessory και να δοθούν οι κατάλληλες απαντήσεις. Επίσης “ακούει” για το event του http server listening προκειμένου να ξεκινήσει να στείλει τα δεδομένα του κάθε εξαρτήματος σε κάθε συσκευή που είναι συνδεδεμένη στον server.

HAPServer.js: Αυτή η κλάση είναι υπεύθυνη για την μεταφορά κάθε εντολής από και προς το Homekit στις iOS συσκευές. Υλοποιεί έναν Evented HTTP server και χρησιμοποιεί κωδικοποίηση δεδομένων ChaCha20 σε συνδυασμό με την Poly1305 στην αποστολή κάθε μηνύματος, μετά την ζευγοποίηση κάθε συσκευής με τον server, ενώ χρησιμοποιεί HTTP/1.1 χωρίς κρυπτογράφηση κατά την ζευγοποίηση. Εμπεριέχονται μέθοδοι για την διαδικασία της ταυτοποίησης με την συσκευή μέσω βημάτων που υποστηρίζονται από το πρωτόκολλο Homekit. Επίσης διαθέτει μεθόδους που αποστέλλουν στην κάθε συσκευή τις λεπτομέρειες των διαθέσιμων αξεσουάρ, ενώ διαθέτει και μεθόδους προκειμένου να προωθεί τα μηνύματα αλλαγής

και ανάγνωσης χαρακτηριστικών στα κατάλληλα αντικείμενα. Τέλος δημιουργεί event κάθε φορά που λαμβάνεται κάποιο μήνυμα, προκειμένου να μπορεί να αναγνωσθεί από κάθε αντικείμενο που έχει ζητήσει να ενημερώνεται.

Advertiser.js: Αυτή η κλάση είναι υπεύθυνη για την εμφάνιση των αξεσουάρ στο τοπικό δίκτυο με την βοήθεια της τεχνολογίας του multicast DNS ή αλλιώς Bonjour. Είναι υπεύθυνη για την ανανέωση των στοιχείων ενός εξαρτήματος στο Bonjour αλλά δεν αποστέλλει ανανεώσεις αυτή η κλάση προς τις συσκευές, απλά εάν ξανά διαβάσουν οι συσκευές τον Bonjour server εξασφαλίζει ότι θα περιέχει τις σωστές. Τέλος είναι υπεύθυνη για την εμφάνιση κάποιου αξεσουάρ διαθέσιμου(online) ή μη προσβάσιμου(offline).

Bridge.js: Αποτελεί την κλάση που αναπαριστά μια ειδική κατηγορία αξεσουάρ, τις γέφυρες(bridges) που έχουν την ιδιότητα να ενσωματώνουν παραπάνω από ένα αξεσουάρ σε μία συσκευή και είναι υπεύθυνες για τον έλεγχο τους. Δεν διαθέτει περαιτέρω μεθόδους, αντίθετα καλεί την κλάση Accessory.js με μια παραπάνω μεταβλητή για την δημιουργία αξεσουάρ τύπου γέφυρας.

Eventedhttp.js: Αυτή η κλάση χρησιμοποιείται για την δημιουργία ενός tcp socket μεταξύ του iOS και του υπολογιστή που τρέχει το nodeJS. Μετά την δημιουργία αυτής της σύνδεσης αποστέλλονται τα μηνύματα που μεταφέρονται μέσω των socket, με event σε κάθε κλάση που περιέχει αντικείμενο του evented http server και έχει ζητήσει να ενημερώνεται για τα μηνύματα αυτά, ενώ παρέχονται παράλληλα και μέθοδοι για την αποστολή μηνυμάτων από το nodejs στο socket. Αυτή η κλάση δεν είναι υπεύθυνη για τις διαδικασίες αυθεντικοποίησης μεταξύ του iOS και του nodeJS αλλά μόνο για την λήψη μηνυμάτων και την διαχείριση της σύνδεσης.

[Πηγή: 9]

5.1.1 Κλάσεις χειρισμού αξεσουάρ

Light_accessory.js: Αυτή η κλάση είναι υπεύθυνη για τον χειρισμό του αξεσουάρ με τύπο ελέγχου λάμπας. Ο τύπος ελέγχου του αξεσουάρ καθορίζεται από το όνομα που έχει δοθεί στο bluetooth module, που υπάρχει πάνω σε κάθε αξεσουάρ. Έτσι τα αξεσουάρ με όνομα LIGHTS ελέγχονται από την συγκεκριμένη κλάση. Μέσα σε αυτή την κλάση δημιουργούνται οι απαραίτητες συνδέσεις για την επίτευξη σύνδεσης με το κάθε bluetooth module, προκειμένου να είναι δυνατή η αποστολή και η λήψη των εντολών μεταξύ των δύο συσκευών. Επίσης αυτή η κλάση είναι υπεύθυνη για την εφαρμογή των αλλαγών που ζητούνται από κάθε iOS συσκευή και για την

ειδοποίηση όλων των συσκευών κάθε φορά που αλλάζει κάποια τιμή ενός χαρακτηριστικού της λάμπας. Τέλος υποστηρίζει την δυνατότητα χειρισμού λαμπών με πολλαπλά χρώματα.

Switch_accessory.js: Αυτή η κλάση είναι υπεύθυνη για τον χειρισμό του αξεσουάρ με τύπο ελέγχου διακόπτη. Ο τύπος ελέγχου του αξεσουάρ καθορίζεται από το όνομα που έχει δοθεί στο bluetooth module που υπάρχει πάνω σε κάθε αξεσουάρ. Έτσι τα αξεσουάρ με όνομα SWITCHES ελέγχονται από την συγκεκριμένη κλάση. Μέσα σε αυτή την κλάση δημιουργούνται οι απαραίτητες συνδέσεις για την επίτευξη σύνδεσης με το κάθε bluetooth module προκειμένου να είναι δυνατή, η αποστολή και η λήψη των εντολών μεταξύ των δύο συσκευών. Επίσης αυτή η κλάση είναι υπεύθυνη για την εφαρμογή των αλλαγών που ζητούνται από κάθε iOS συσκευή και για την ειδοποίηση όλων των συσκευών κάθε φορά που αλλάζει κάποια τιμή ενός χαρακτηριστικού του διακόπτη.

Alarco_Alarm_accessory.js: Αυτή η κλάση είναι υπεύθυνη για τον χειρισμό του αξεσουάρ με τύπο ελέγχου διακόπτη. Ο τύπος ελέγχου του αξεσουάρ καθορίζεται από το όνομα που έχει δοθεί στο bluetooth module που υπάρχει πάνω σε κάθε αξεσουάρ. Έτσι τα αξεσουάρ με όνομα ALARM ελέγχονται από την συγκεκριμένη κλάση. Μέσα σε αυτή την κλάση δημιουργούνται οι απαραίτητες συνδέσεις για την επίτευξη σύνδεσης με το κάθε bluetooth module, προκειμένου να είναι δυνατή η αποστολή και η λήψη των εντολών μεταξύ των δύο συσκευών. Επίσης αυτή η κλάση είναι υπεύθυνη για την εφαρμογή των αλλαγών που ζητούνται από κάθε iOS συσκευή και για την ειδοποίηση όλων των συσκευών κάθε φορά που αλλάζει κάποια τιμή ενός χαρακτηριστικού του συναγερμού. Οι λειτουργίες που υποστηρίζονται όσον αφορά τις διαφορετικές καταστάσεις που μπορεί να έχει ο συναγερμός είναι: α. Οπλισμός των μη-ενεργών περιοχών, β. Οπλισμός όλων των περιοχών, γ. Οπλισμός των νυχτερινών μη προσβάσιμων περιοχών, δ. Αφοπλισμός. Τέλος ειδοποιεί τον χρήστη με push notification στην περίπτωση που ενεργοποιηθεί η σειρήνα του συναγερμού.

5.2 Multicast DNS / Bonjour και ο πρόγονος τους

Στην δικτύωση υπολογιστών, το multicast Domain Name System(mDNS) αντιστοιχίζει ονόματα υπολογιστών σε IP διευθύνσεις, μέσα σε μικρά δίκτυα που δεν

εμπεριέχουν τοπικούς DNS server. Είναι μια υπηρεσία που δεν απαιτεί καμία διαμόρφωση, χρησιμοποιώντας ουσιαστικά τις ίδιες συμβάσεις προγραμματισμού, σχηματισμούς πακέτων και λειτουργικές σημασιολογίες όπως το unicast Domain Name System(DNS). Παρόλο που έχει σχεδιαστεί από τον Stuart Cheshire για να λειτουργεί αυτόνομα, μπορεί να συνεργαστεί με unicast DNS servers.
Το mDNS έχει εκδοθεί μέσα στην δικτυακή οδηγία με αριθμό RFC 6762, χρησιμοποιεί πακέτα IP multicast User Datagram Protocol(UDP) και έχει εφαρμοστεί στις υπηρεσίες του Bonjour της Apple και του nss-mdns της Linux.
Το mDNS μπορεί να χρησιμοποιηθεί σε συνδυασμό με το DNS Service Discovery(DNS-SD), μια συνοδευτική τεχνική που δεν απαιτεί διαμόρφωση που περιγράφεται στην δικτυακή οδηγία RFC 6763.

5.2.1 Περιγραφή πρωτοκόλλου

Όταν ένας πελάτης mDNS χρειάζεται να βρει την IP ενός ονόματος server, στέλνει ένα IP multicast μήνυμα ερώτησης που ζητά από τον server με αυτό το όνομα να ταυτοποιήσει τον εαυτό του. Αυτό το μηχάνημα προορισμού τότε απαντά με ένα μήνυμα το οποίο περιέχει την IP διεύθυνση του. Όλα τα μηχανήματα στο υπο-δίκτυο μπορούν να χρησιμοποιήσουν αυτή την λειτουργία για να αναβαθμίσουν τους καταλόγους mDNS που διαθέτουν.

Οποιοσδήποτε υπολογιστής μπορεί να αποδεσμεύσει τον έλεγχο ενός domain name στέλνοντας ένα πακέτο απάντησης με μηδενικό χρόνο ζωής(TTL).

Εξ ορισμού, το mDNS είναι υπεύθυνο για την εύρεση των IP διευθύνσεων μόνο για domain name που έχουν κατάληξη “.local”. Αυτό μπορεί να προκαλέσει προβλήματα αν το domain αυτό περιέχει υπολογιστές που δεν υλοποιούν το πρωτόκολλο mDNS, αλλά μπορούν να βρεθούν από έναν συμβατικό unicast DNS server. Επίλυση τέτοιων προβλημάτων απαιτεί αναδιαμόρφωση στο δίκτυο που παραβιάζει τον στόχο της απουσίας διαμόρφωσης.

5.2.2 Δομή πακέτου

Το mDNS πλαίσιο Ethernet είναι ένα multicast UDP πακέτο προς:

- Διευθύνσεις MAC είτε IPv4 είτε IPv6,
- IPv4 διευθύνσεις ή IPv6 διευθύνσεις,

- την UDP πόρτα 5353.

Η δομή του πακέτου του βασίζεται στη μορφή του DNS πακέτου. Αποτελείται από δύο μέρη - την κεφαλίδα και τα δεδομένα.

offset (bytes)	0	1
0	ID = 0x0000	
2	Flags	
4	QDCOUNT	
6	ANCOUNT	
8	NSCOUNT	
10	ARCOUNT	
12	Data	

Πίνακας 5.1 : Δομή πακέτου mDNS

Η λέξη FLAGS θα είναι συνήθως 00 00 για ερωτήματα και 84 00 για απαντήσεις. Τα δεδομένα κάθε πακέτου αρχίζουν μαζί με το FQDN(fully qualified domain name) προς εύρεση και τελειώνουν με 2 σημαίες των 2 byte που δείχνουν το QTYPE(00 01 για την εύρεση διεύθυνσης server) και το QCLASS(00 01 για το Internet):

- Τα δεδομένα ενός πακέτου τύπου εύρεσης δεν περιέχουν άλλες πληροφορίες.
- Στα δεδομένα του πακέτου απάντησης, το FQDN ακολουθείται από την IPv4 διεύθυνση του server προς εύρεση, την IPv6 διεύθυνση και τα name section records

Το FQDN ορίζεται από μια λίστα συνδεδεμένων συμβολοσειρών, ξεκινώντας με το όνομα του εξυπηρετητή και τελειώνοντας με το top-level domain(TLD). Κάθε τέτοια συμβολοσειρά αποτελείται από ένα byte που δείχνει το μήκος της συμβολοσειράς και ακολουθείται από αυτά τα UTF-8 byte. Το TLD ακολουθείται από μια null συμβολοσειρά (hex 00) που συμβολίζει τον τερματισμό του FQDN.

Η IPv4 μέρος του πακέτου του τελικού εξυπηρετητή αποτελείται από:

- ένα πεδίο 2-byte για τον τύπο(hex 00 00 για ένα A record),
- ένα πεδίο 2-byte για την κλάση(hex 80 01 για την κλάση IN με το bit καθαρισμού cache ενεργοποιημένο),

- έναν ακέραιο με προσήμανση 32-bit για το πεδίο Time To Live (TTL) (σε δευτερόλεπτα),
- ένα πεδίο 2-byte για το μήκος(hex 00 04 για μια διεύθυνση IPv4 4 byte)
- τα 4 IPv4 byte της διεύθυνσης

Το IPv6 μέρος του πακέτου αποτελείται από:

- ένα link 2-byte που δείχνει την μετατόπιση του ονόματος εξυπηρετητή(hex C0 0C),
- ένα πεδίο 2-byte για τον τύπο(hex 00 1C για τον τύπο AAAA),
- ένα πεδίο 2-byte για την κλάση(hex 80 01 για την κλάση IN με ενεργοποιημένο το bit εκκαθάρισης cache),
- ένα προσημασμένο ακέραιο 32-bit για το πεδίο Time To Live (TTL),
- ένα πεδίο 2-byte για το μήκος (hex 00 10 για μια διεύθυνση 16-byte IPv6), και
- τα 16 byte της IPv6 διεύθυνσης.

Το πεδίο του ονόματος του πακέτου αποτελείται από:

- ένα σύνδεσμο 2-byte που δείχνει την μετατόπιση του ονόματος εξυπηρετητή(hex C0 0C),
- ένα πεδίο 2-byte που δείχνει τον τύπο (hex 00 2f για τον τύπο NSEC),
- ένα πεδίο 2-byte που δείχνει την κλάση (hex 80 01 για την κλάση IN με ενεργοποιημένο το bit εκκαθάρισης cache),
- ένα προσημασμένο ακέραιο 32-bit για το πεδίο Time To Live (TTL),
- ένα πεδίο 2-byte για το μήκος(hex 00 08 για μια καταχώρηση 8-byte NSEC), και
- τα 8 block και bitmap bytes.

Σε ένα κανονικό DNS μήνυμα το πρώτο byte κάθε πεδίου κλάσης είναι πάντα 0x00, στο mDNS το πρώτο bit του byte αυτού μπορεί να αλλάξει, θέτοντας την αξία του στο 0x80. Αυτό είναι το bit για τη εκκαθάριση της κλάσης δεν πρέπει να ερμηνεύεται ως μέρος του αριθμού της κλάσης.

Παράδειγμα

Δοκιμάζοντας να κάνουμε ping στον εξυπηρετητή appletv.local, θα προκαλούσε έναν πελάτη mDNS να μεταδώσει δημόσια το ακόλουθο UDP πακέτο:
00 00 00 00 00 01 00 00 00 00 00 00 07 61 70 70

6c 65 74 76 05 6c 6f 63 61 6c 00 00 01 00 01

Και τα 6 πεδία της κεφαλής ισούνται με μηδέν (00 00) εκτός του QDCOUNT, που ισούται με ένα(00 01). Τα δεδομένα ξεκινούν με τους επτά χαρακτήρες του appletv host name (hex 07 61 70 70 6c 65 74 76) ακολουθούμενα από τη συμβολοσειρά των πέντε χαρακτήρων του local domain (hex 05 6c 6f 63 61 6c) και την απαιτούμενη null συμβολοσειρά (00). Ολόκληρο το FQDN ακολουθείται από την διεύθυνση εξυπηρετητή τη σημαία QTYPE (hex 00 01) και τη σημαία internet QCLASS (00 01).

Ο εξυπηρετητής appletv.local θα απαντούσε στέλνοντας δημόσια το πακέτο απάντησης mDNS. Για παράδειγμα:

00 00 84 00 00 00 00 01 00 00 00 02 07 61 70 70
6c 65 74 76 05 6c 6f 63 61 6c 00 00 01 80 01 00
00 78 00 00 04 99 6d 07 5a c0 0c 00 1c 80 01 00
00 78 00 00 10 fe 80 00 00 00 00 00 00 02 23 32
ff fe b1 21 52 c0 0c 00 2f 80 01 00 00 78 00 00
08 c0 0c 00 04 40 00 00 08

Στην κεφαλίδα του, τα μη μηδενικά πεδία είναι η λέξη Σημαίες(84 00), η λέξη ANCOUNT (00 01), και η λέξη ARCOUNT (00 02). Τα δεδομένα ξεκινούν με το FQDN (hex 07 61 70 70 6c 65 74 76 05 6c 6f 63 61 6c 00 για το appletv.local), ακολουθούμενα από τις πληροφορίες DNS αυτού του εξυπηρετητή:

- ο κωδικός τύπου διεύθυνσης A/IPv4(hex 00 01),
- ο κωδικός κλάσης IPv4 (hex 80 01),
- το IPv4 TTL (hex 00 00 78 00 for 30720 δευτερόλεπτα),
- το μήκος του IPv4 (hex 00 04),
- τα τέσσερα byte IPv4 διεύθυνσης (hex 99 6D 07 5A, ή 153.109.7.90 σε μορφή dotted-decimal),
- η μετατόπιση του FQDN (hex C0 0C για το byte 12),
- ο κωδικός τύπου διεύθυνσης AAAA/IPv6 (hex 00 1C),
- ο κωδικός κλάσης IPv6 (hex 80 01),
- το IPv6 TTL(πάλι hex 00 00 78 00),
- το μήκος του IPv6 (hex 00 10),
- τα 16 byte της διεύθυνσης IPv6 (hex FE 80 00 00 00 00 00 00 02 23 32 FF FE B1 21 52),

- η μετατόπιση του FQDN (hex C0 0C για το byte 12),
- ο κωδικός τύπου NSEC(hex 00 2F),
- ο κωδικός κλάσης NSEC (hex 80 01),
- το NSEC TTL (again hex 00 00 78 00),
- το μήκος του NSEC (hex 00 08, για ένα πεδίο ονόματος μήκους 8 byte), και
- τα 8 NSEC block και bitmap bytes (hex C0 0C 00 04 40 00 00 08)

5.2.3 AppleTalk®

Ο πρόγονος της τεχνολογίας του mDNS ξεκίνησε τη ζωή του ως ένα χαρακτηριστικό αποκλειστικά των Apple Computers®. Η δημιουργία του mDNS είχε σαν στόχο την δημιουργία ενός πρωτοκόλλου το οποίο θα μπορούσε να λειτουργήσει μέσα σε μια στοίβα IP πακέτων και θα αντικαθιστούσε το AppleTalk NBP(Name Binding Protocol). Ένα από τα χαρακτηριστικά του Apple Talk ήταν η λειτουργία του δικτύου χωρίς την απαίτηση καμίας χειροκίνητης διαμόρφωσης του δικτύου ή την χρήση κάποιας υπηρεσίας για την διαμόρφωση του. Μια από τις βασικές λειτουργίες του AppleTalk ήταν η σύνδεση των ονομάτων υπολογιστών σε διευθύνσεις δικτύου, παρόλο που οι περισσότεροι χρήστες της τεχνολογίας το είχαν αντιστοιχήσει με την δυνατότητα να εμφανίζει όλες τις διαθέσιμες τοπικές συσκευές σε ένα δίκτυο, εξαιτίας του AppleTalk Chooser, που σου έδινε την δυνατότητα να επιλέξεις ένα από τους διαθέσιμους υπολογιστές. Επίσης το συγκεκριμένο πρωτόκολλο δεν εμφάνιζε απλά τους διαθέσιμους εξυπηρετητές αλλά τις υπηρεσίες που υποστήριζε ο κάθε εξυπηρετητής, όπως η ftp μεταφορά, η εκτύπωση ή και το http serving. Ακόμα μια σημαντική ευελιξία του συστήματος AppleTalk ήταν η δυνατότητα να απαντάει κάθε υπηρεσία μιας συσκευής με μια συγκεκριμένη διεύθυνση για εκείνη την υπηρεσία και όχι με την διεύθυνση ολόκληρου του εξυπηρετητή και να αναγκάζει τον πελάτη(client) να προσδιορίζει εκείνος την πόρτα στην οποία βρίσκεται η υπηρεσία που ήθελε να χρησιμοποιήσει αναγκάζοντας τον να ξέρει από πριν την συγκεκριμένη υπηρεσία. Στο πρωτόκολλο AppleTalk κάθε υπηρεσία ενός εξυπηρετητή ακούει σε μια διαφορετική πρίζα(socket ή όπως την ξέρουμε από το IP πρωτόκολλο πόρτα-port) και διαφημίζει τον εαυτό της στο δίκτυο ως ξεχωριστή NBP οντότητα. Όταν ένας πελάτης(client) ζητά μια υπηρεσία από το AppleTalk δίκτυο ως απάντηση δίνονταν και το δίκτυο, υποδίκτυο, εξυπηρετητής στο υποδίκτυο αλλά και η πρίζα(socket) στην

οποία βρίσκεται η υπηρεσία. Επιπρόσθετα το AppleTalk διαθέτει τυποποιημένα ονόματα υπηρεσιών που ακολουθούν την σύμβαση:

Όπου το όνομα είναι το όνομα της υπηρεσίας που είναι ορατό στον χρήστη, ο τύπος είναι ένα αναγνωριστικό που καθορίζει το πρωτόκολλο της υπηρεσίας και τα σημασιολογικά. Μπορούμε να το προσδιορίσουμε και ως μια περιγραφή της τελικής δράσης που πραγματοποιεί μια υπηρεσία(π.χ.: εκτύπωση). Από μια καθαρά τεχνική πλευρά ο τύπος μια υπηρεσίας προσδιορίζει το σημασιολογικό πρωτόκολλο της εφαρμογής που χρησιμοποιεί η υπηρεσία. Ενώ η ζώνη είναι μια μέθοδος οργανωτικής ή γεωγραφικής ένωσης πολλαπλών υπηρεσιών. Μια συνηθισμένη ονοματολογία ζωνών ήταν το “Πωλήσεις”, “Γραφείο 3”, “Κτίριο 1”. Το αντίστοιχο στο πρωτόκολλο DNS θα ήταν τα subdomain όπως το “Sales.example.com”.

5.2.4 DNS-SD(DNS Service Discovery)

Το DNS-SD επιτρέπει σε πελάτες(clients) να βρίσκουν μια ονοματική λίστα με διαθέσιμες υπηρεσίες, αφού δώσουν έναν τύπο υπηρεσίας, μπορούν να αντιστοιχίζουν αυτές τις υπηρεσίες σε ονόματα εξυπηρετητών χρησιμοποιώντας συνηθισμένες αναζητήσεις DNS. Η προδιαγραφή αυτή είναι συμβατή με το υπάρχον λογισμικό εξυπηρετητών και πελατών unicast DNS, αλλά δουλεύει το ίδιο καλά και με multicast DNS σε ένα περιβάλλον μηδενικής διαμόρφωσης. Κάθε υπηρεσία περιγράφεται με μια DNS SRV και μια DNS TXT καταχώρηση. Ένας πελάτης βρίσκει την λίστα των διαθέσιμων υπηρεσιών για ένα συγκεκριμένο τύπο υπηρεσίας ζητώντας το αρχείο DNS PTR του ονόματος του συγκεκριμένου τύπου υπηρεσίας. Ο εξυπηρετητής επιστρέφει μηδέν ή περισσότερα ονόματα με την μορφή “<Service>.<Domain>”, που αντιστοιχίζεται σε ένα ζευγάρι καταχώρησης SRV/TXT.

Η SRV καταχώρηση αντιστοιχίζεται σε ένα domain name που περιέχει την υπηρεσία, ενώ το TXT μπορεί να περιέχει παραμέτρους διαμόρφωσης για την συγκεκριμένη υπηρεσία. Ένας πελάτης μπορεί να βρεί την καταχώρηση A/AAAA για το domain name και να συνδεθεί στην υπηρεσία.

5.2.5 Ιστορικό

Το 1997 ο Stuart Cheshire πρότεινε την προσαρμογή του καλό-σχεδιασμένου Name Binding Protocol της Apple στα IP δίκτυα για την επίλυση της ανυπαρξίας δυνατοτήτων εύρεσης υπηρεσιών. Ο Cheshire στην συνέχεια προσελήφθη στην Apple και έγραψε τα πρόχειρα προτάσεων προς το IETF για το multicast DNS και τη βασισμένη στο DNS Service Discovery, υποστηρίζοντας την μετάβαση από το AppleTalk στην IP δικτύωση. Το 2002, η Apple ανακοίνωσε την εφαρμογή και των δύο πρωτοκόλλων κάτω από το όνομα Rendezvous(αργότερα ονομαζόμενο Bonjour), που συμπεριλήφθηκε στο Mac OS X 10.2 και αντικατέστησε το Service Location Protocol που χρησιμοποιούσε η 10.1. Το 2013, οι προτάσεις επικυρώθηκαν στα RFC 6762 και RFC 6763.[19]

DNS-SD με multi cast

Το Multicast DNS (mDNS) είναι ένα πρωτόκολλο που χρησιμοποιεί πακέτα παρόμοια με αυτά του unicast DNS αλλά στέλνονται μέσω unicast για την αντιστοίχηση ονομάτων εξυπηρετητών. Κάθε εξυπηρετητής “ακούει” στην mDNS πόρτα, 5353, και απαντάει σε αιτήματα για την DNS καταχώρηση του .local ονόματος του(πχ το A, AAAA, CNAME) με την IP διεύθυνση του. Όταν ένας πελάτης mDNS πρεέπει να αντιστοιχείσει το τοπικό όνομα ενός εξυπηρετητή σε μια IP διεύθυνση, στέλνει μια DNS αίτηση για το όνομα αυτό στη γνωστή unicast διεύθυνση και ο υπολογιστής με την αντίστοιχη καταχώρηση A/AAAA απαντάει με την IP διεύθυνση του. Η διεύθυνση mDNS multicast είναι 224.0.0.251 στο IPv4 και ff02::fb για την IPv6 link-local διευθυνσιοδότηση.

Οι αιτήσεις DNS service discovery (DNS-SD) μπορούν επίσης να σταλούν μέσω ενός συνδέσμου multicast, και μπορούν να συνδυαστούν με το mDNS για την επίτευξη μηδενικής διαμόρφωσης DNS-SD. Χρησιμοποιεί και πάλι DNS PTR, SRV, TXT καταχωρήσεις για την δημοσίευση αντικειμένων(instances) με τύπους υπηρεσιών, domain names για αυτά τα αντικείμενα, και προαιρετικά παραμέτρους διαμόρφωσης για την σύνδεση σε αυτά τα αντικείμενα. Αλλά οι SRV καταχωρήσεις μπορούν να αντιστοιχιστούν τώρα σε multicastable .local domain names, τα οποία το mDNS μπορεί να αντιστοιχίσει σε τοπικές IP διευθύνσεις.

Υποστήριξη

Το DNS-SD χρησιμοποιείται από προϊόντα της Apple, τους περισσότερους δικτυακούς εκτυπωτές, πολλές διανομές του Linux συμπεριλαμβανομένων των Debian και Ubuntu, και έναν αριθμό από προϊόντα τρίτων κατασκευαστών για διάφορα λειτουργικά συστήματα. Για παράδειγμα, πολλές δικτυακές εφαρμογές του OS X γραμμένες από την Apple, συμπεριλαμβανομένων των Safari, iChat και του Messages, μπορεί να χρησιμοποιήσουν DNS-SD για να βρουν κοντινούς servers και p2p πελάτες. Στα Windows, το λειτουργικό σύστμα υποστηρίζει το DNS-SD τουλάχιστον στα Windows 10 για εφαρμογές γραμμένες σε JavaScript και άλλες γλώσσες που μπορεί να προστεθούν σύντομα. Ξεχωριστές εφαρμογές μπορεί να περιλαμβάνουν δική τους υποστήριξη σε παλαιότερες εκδόσεις του λειτουργικού συστήματος, όπως οι περισσότερες εφαρμογές instant messaging και VoIP clients στα Windows υποστηρίζουν DNS-SD. Κάποιες διανομές Unix, BSD και Linux επίσης υποστηρίζουν DNS-SD. Για παράδειγμα, το Ubuntu περιλαμβάνει το Avahi, μια υλοποίηση mDNS / DNS-SD στη βασική του διανομή.

Κεφάλαιο 6

Εφαρμογή iOS

Πρόκειται για μια εφαρμογή η οποία είναι το κέντρο ελέγχου όλων των αξεσουάρ μας, γραμμένη στην native γλώσσα της Apple - την Objective C - και χρησιμοποιώντας το μοντέλο συγγραφής κώδικα MVC(Model - View - Controller). Χρησιμοποιεί το framework Homekit για την επικοινωνία με τα εικονικά αξεσουάρ που βρίσκονται εγγεγραμμένα στην εφαρμογή μας στην πλατφόρμα NodeJS.
Γραφικά πρόκειται για μια εξαιρετικά βασική εφαρμογή χωρίς πολλά οπτικά εφέ, καθώς δεν υπήρχε χρόνος για την περαιτέρω ανάπτυξη της εφαρμογής σε γραφικό επίπεδο.

Έχει την δυνατότητα να ελέγξει κάθε είδους εξαρτήματα που υποστηρίζονται από το πρωτόκολλο Homekit, είτε αυτά είναι μερικά από τα εξαρτήματα που έχουμε φτιάξει ως εξαρτήματα για αυτή την εργασία είτε όχι. Πάρθηκε η απόφαση αυτή προκειμένου να μπορέσει η εφαρμογή να έχει την δυνατότητα μελλοντικής επέκτασης της με εξαρτήματα που αυτή την στιγμή δεν έχουμε σκεφτεί να κατασκευάσουμε.

Έχει την δυνατότητα να παρέχει ειδοποιήσεις στον χρήστη για αλλαγές στην κατάσταση των αξεσουάρ του, σε όποια εφαρμογή και αν βρίσκεται με έναν ήχο/δόνηση και ένα μήνυμα το οποίο είναι σαφές και εμπεριέχει όλες τις απαραίτητες πληροφορίες για να γίνει κατανοητό στον χρήστη ποια ακριβώς αλλαγή πραγματοποιήθηκε.

Ακόμα έχει την δυνατότητα μέσω της ηλεκτρονικής βοηθού “Siri” να ενεργοποιεί, να απενεργοποιεί, αλλά και να πραγματοποιεί κάθε υποστηριζόμενη από το εξάρτημα δράση, χρησιμοποιώντας μόνο την φωνή και με την χρήση φυσικής γλώσσας. Προς το παρόν αυτή η λειτουργία δεν υποστηρίζει την ελληνική γλώσσα, αλλά μόνο γλώσσες όπως η Αγγλική, Γερμανική, Ιταλική, Γαλλική και Τούρκικη. Αναμένουμε όμως πολύ σύντομα να είναι διαθέσιμη και στα ελληνικά όπως μας το έκανε σαφές η Apple όταν πριν λίγους μήνες με την αναβάθμιση στο λειτουργικό iOS 9.1, για την πρώτη της δοκιμαστική έκδοση είχε ξεχαστεί ως επιλογή η ελληνική γλώσσα για τον βοηθό Siri - κάτι που γρήγορα επιδιορθώθηκε.

Επίσης έχει την δυνατότητα να εκτελεί πολύπλοκες εργασίες όπως η δημιουργία χρονοδιακοπτών για την πραγματοποίηση αλλαγών στην κατάσταση εξαρτημάτων. Να δημιουργεί group ενεργειών έτσι ώστε να αυτοματοποιεί ορισμένες ενέργειες χωρίς να πρέπει να εκτελεί την κάθε μια από το αντίστοιχο menu. Να δημιουργεί δωμάτια στα οποία να εντάσσει εξαρτήματα για να κάνει πιο εύκολη την αναγνώριση τους αλλά και να κάνει πιο εύκολες κινήσεις όπως η ενεργοποίηση όλων των λαμπών στο σαλόνι ή στην κουζίνα. Να ορίζει ζώνες οι οποίες αποτελούν group από δωμάτια. Και τέλος έχει την δυνατότητα να παραχωρεί την λειτουργία των εξαρτημάτων του σε άλλους χρήστες, απλά εισάγοντας τον λογαριασμό iCloud ενός άλλου χρήστη οπότε και αποστέλλεται μια πρόσκληση προς τον χρήστη εκείνο, την οποία πρέπει να αποδεχθεί προκειμένου να μπορεί να ελέγξει τα εξαρτήματα μας. Απαραίτητη προϋπόθεση βέβαια για να συμβεί αυτό, είναι να διαθέτει και ο άλλος χρήστης την εφαρμογή μας εγκατεστημένη στην συσκευή του.

Ο τρόπος λειτουργίας της είναι ο ακόλουθος: Ζητείται από τον χρήστη εξ' αρχής να ορίσει ένα νέο εικονικό "Σπίτι". Αυτό θα αποτελέσει την δομή η οποία θα εμπεριέχει μέσα της κάθε μια από τις δυνατότητες που θα έχει ο χρήστης στην διάθεση του και θα εμπεριέχει και κάθε εξάρτημα που θα προσθέτουμε. Θα ενημερώνεται αυτή η δομή κάθε φορά που υπάρχει μια ενημέρωση στα εξαρτήματα μας από την NodeJS εφαρμογή. Στην συνέχεια μπορεί να πατήσει το κουμπί "+" στην επάνω δεξιά γωνία της οθόνης, το οποίο ξεκινάει την διαδικασία αναζήτησης εξαρτημάτων συμβατών με το πρωτόκολλο HomeKit. Επιλέγει το(α) εξάρτημα(τα) που θέλει να προσθέσει και στην συνέχεια του ζητείται να εισάγει έναν κωδικό, ο οποίος θα αναγράφεται είτε πάνω στην συσκευή την ίδια είτε πάνω στον υπολογιστή που τρέχει την εφαρμογή NodeJS, και το οποίο μπορεί και να σκαναριστεί για διευκόλυνση του χρήστη μέσω της κάμερας της συσκευής, αντί να εισαχθεί χειροκίνητα, από ένα ειδικό αυτοκόλλητο που θα βρίσκεται πάνω στην συσκευή. Τέλος το εξάρτημα έχει προστεθεί στο "Σπίτι" του χρήστη και πλέον έχει όλες τις παραπάνω αναφερθείσες δυνατότητες.

6.1 Ανάλυση δομής

Ο γονικός φάκελος της εφαρμογής ονομάζεται HMCatalog και διαθέτει όλους τους υποφακέλους της εφαρμογής που περιέχουν πηγαίο κώδικα. Σε αυτόν περιέχονται τα εξής αρχεία:

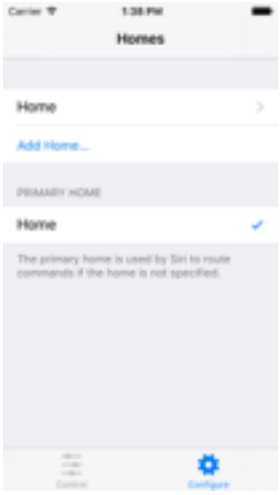
- HomeKitTableViewController: Αυτό είναι το αρχείο που αποτελεί την σπονδυλική στήλη του όλου project μας. Όλες οι οθόνες της εφαρμογής αποτελούνται από πίνακες και αυτό το αρχείο περιέχει την κλάση HomeKitTableViewController, η οποία είναι η υπερκλάση από την οποία υιοθετούν τα χαρακτηριστικά τους όλες οι κλάσεις, οι οποίες εμφανίζουν τα δεδομένα τους στην οθόνη. Αυτή η κλάση περιέχει μέσα της μεθόδους προκειμένου να προκαλεί ανανέωση των δεδομένων στους πίνακες, που είναι παιδιά της μετά από κάθε ανανέωση στα δεδομένα των εξαρτημάτων του σπιτιού. Προκειμένου να πραγματοποιεί τέτοιες ενέργειες εγγράφει τον εαυτό της ως listener στα event τύπου "HomeStoreDidUpdateHomesNotification", μέσω του αντικειμένου NSNotificationCenter, το οποίο είναι ένα αντικείμενο που ενημερώνεται από το Homekit framework, κάθε φορά που αλλάζει κάτι στα εξαρτήματα του σπιτιού και το οποίο χρησιμοποιείται εκτενώς σε όλο το λειτουργικό σύστημα, από εφαρμογές είτε τρίτων είτε εφαρμογές της Apple ως μια διευκόλυνση στην επικοινωνία μεταξύ εφαρμογών. Κάθε εφαρμογή έχει ένα αντίγραφο του συγκεκριμένου αντικειμένου και μπορεί είτε να ακούει για event είτε να ενεργοποιεί η ίδια δικά της event για να ενημερώνει άλλα μέρη της εφαρμογής εξωτερικά της κάθε κλάσης. Το HomeKitTableViewController το ίδιο είναι ένα παιδί της κλάσης UITableViewController το οποίο περιέχει μεθόδους και αντικείμενα για τον έλεγχο του γραφικού περιβάλλοντος πινάκων.
- Main.Storyboard: Αυτή είναι μια δομή η οποία είναι ικανή να περιέχει μέσα της πολλαπλά γραφικά περιβάλλοντα. Μπορεί να κρατά πληροφορίες για το ποιες οθόνες εμφανίζονται όταν ενεργοποιούνται event όπως το click σε ένα κουμπί ή σε μια καταχώρηση ενός πίνακα. Ενώ έχει επίσης την δυνατότητα να αναθέτει κλάσεις σε οθόνες, προκειμένου να μπορούν να ελεγχθούν από εκείνη την κλάση. Τέλος είναι ο μόνος τρόπος γραφικής σχεδίασης μια οθόνης, με δυνατότητες για αυτόματη προσαρμογή σε πολλαπλές διαστάσεις οθονών συσκευών.
- AppDelegate: Είναι η κλάση που εκτελείται κατά την εκκίνηση της εφαρμογής, η main όπως θα την ονόμαζαν άλλες γλώσσες προγραμματισμού. Περιέχει μεθόδους που μπορούν να γίνουν override, προκειμένου να εκτελεστεί κώδικας πριν την εκκίνηση του εκτελέσιμου της εφαρμογής, μετά από αυτή και κατά τις διαδικασίες που η εφαρμογή θα μπει στο παρασκήνιο.

- Localizable.strings: Είναι το αρχείο που κρατά τις μεταφράσεις σε όλες τις γλώσσες που υποστηρίζει η εφαρμογή για την μετάφραση των λεκτικών της γραφικής διεπαφής.
- Homes folder: Φάκελος που περιέχει τις κλάσεις που χρησιμοποιούνται από τις γραφικές διεπαφές που εμφανίζουν τα διαθέσιμα “σπίτια” που έχουν εγγραφεί στην συσκευή.
- Rooms folder: Φάκελος που περιέχει τις κλάσεις που χρησιμοποιούνται από τις γραφικές διεπαφές που εμφανίζουν τα διαθέσιμα δωμάτια που έχουν εγγραφεί στο “Σπίτι”.
- Zones folder: Φάκελος που περιέχει τις κλάσεις που χρησιμοποιούνται από τις γραφικές διεπαφές που εμφανίζουν τις διαθέσιμες ζώνες που έχουν εγγραφεί στο “Σπίτι”.
- Action Sets folder: Φάκελος που περιέχει τις κλάσεις που χρησιμοποιούνται από τις γραφικές διεπαφές που εμφανίζουν τα διαθέσιμα group ενεργειών που έχουν εγγραφεί στο “Σπίτι”.
- Triggers folder: Φάκελος που περιέχει τις κλάσεις που χρησιμοποιούνται από τις γραφικές διεπαφές που δίνουν την δυνατότητα δημιουργίας και εμφάνισης των χρονοδιακοπτών που έχουν εγγραφεί στο “Σπίτι”.
- Service Groups folder: Φάκελος που περιέχει τις κλάσεις που χρησιμοποιούνται από τις γραφικές διεπαφές που εμφανίζουν τα διαθέσιμα group υπηρεσιών που έχουν εγγραφεί στο “Σπίτι”.
- Utilities folder: Διάφορες κλάσεις που αναλαμβάνουν να κατηγοριοποιούν και να δημιουργούν μηνύματα για κάθε πιθανό event που μπορεί να παραχθεί από την ανανέωση ενός εξαρτήματος.

Και τώρα θα εξετάσουμε αναλυτικά όλους τους φακέλους και τις κλάσεις που διαθέτουν.

6.1.1 Homes Folder

- HomeListConfigurationViewController: Αυτή η κλάση εμπεριέχει τον κώδικα ο οποίος κατασκευάζει τον πίνακα, ο οποίος περιέχει τα αποθηκευμένα “Σπίτια” αλλά και δίνει την δυνατότητα να δημιουργηθούν περισσότερα “Σπίτια”. Επίσης ο χρήστης μέσω αυτής της οθόνης έχει την δυνατότητα να σβήσει κάποιο από τα αποθηκευμένα σπίτια που έχει δημιουργήσει αλλά και να θέσει το πρωτεύον σπίτι της επιλογής του, προκειμένου να μπορεί να ελέγξει τα εξαρτήματα του σπιτιού αυτού, μέσω της ψηφιακής βοηθού Siri. Την κλάση αυτή την χρησιμοποιούμε στο tab configure.

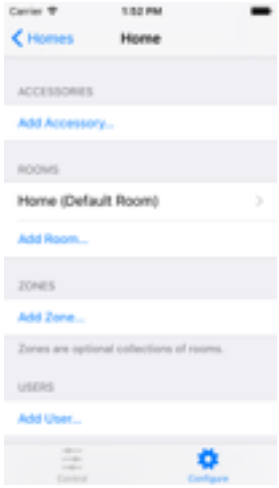


Εικόνα 6.1 : Αρχικό παράθυρο στο tab configure

- HomeListViewController: Αυτή η κλάση εμπεριέχει τον κώδικα ο οποίος κατασκευάζει τον πίνακα, ο οποίος δείχνει όλα τα διαθέσιμα σπίτια, χωρίς καμία δυνατότητα παραμετροποίησης τους. Την χρησιμοποιούμε στο tab “Control” μόνο για την επιλογή του σπιτιού του οποίου θέλουμε να δούμε τα διαθέσιμα εξαρτήματα.

Εικόνα 6.2 : Αρχικό παράθυρο στο tab control

- HomeViewController: Αυτή η κλάση εμπεριέχει κώδικα ο οποίος δημιουργεί τον πίνακα που επιτρέπει την παραμετροποίηση ενός σπιτιού και επιτρέπει την προσθήκη νέων εξαρτημάτων, νέων δωματίων, νέων ζωνών, νέων χρηστών, νέων group πολλαπλών ενεργειών, νέων χρονοδιακοπτών και νέων service group. Αυτή η κλάση δεν είναι υπεύθυνη για την πραγματοποίηση όλων αυτών των ενεργειών, αλλά είναι υπεύθυνη μόνο για την εμφάνιση των υπαρχόντων ενεργειών και εξαρτημάτων και δίνει την δυνατότητα διαγραφής κάθε διαθέσιμης ενέργειας. Την δημιουργία και την παραμετροποίηση των ενεργειών αυτών την αναθέτει σε άλλες κλάσεις που θα αναφερθούν παρακάτω.



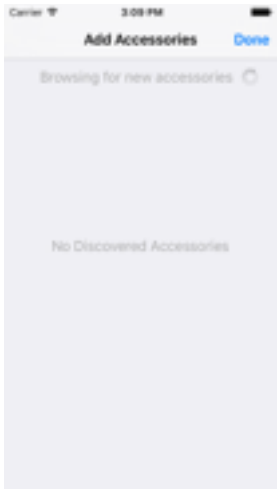
Εικόνα 6.3 : Αρχικό παράθυρο στο tab configure, μετά την επιλογή σπιτιού

- HomeStore: Αυτή η κλάση είναι μια βασική κλάση του όλου μας project καθώς είναι υπεύθυνη για την επικοινωνία της εφαρμογής μας με αντικείμενα τύπου HMHome και συνεπώς είναι υπεύθυνη για την επικοινωνία της εφαρμογής μας με το Homekit framework. Αναλαμβάνει να είναι ο διαχειριστής των “Σπιτιών” που είναι εγγεγραμμένα στο homekit πρωτόκολλο και να ενημερώνει για τυχόν αλλαγές στο κάθε “Σπίτι”, μέσω ειδοποιήσεων που δημιουργεί μέσω του αντικειμένου NSNotificationCenter και για τις οποίες υπάρχουν διάσπαρτοι listeners μέσα στην εφαρμογή μας, προκειμένου να αναβαθμίζονται τα στοιχεία που αλλάζουν με κάθε ενημέρωση. Επίσης αναλαμβάνει να δημιουργεί μια ουρά για κάθε request αλλαγής του “Σπιτιού” που μας ενδιαφέρει, προκειμένου να μην δημιουργούμε προβλήματα με αλλαγές σε ρυθμίσεις εξαρτημάτων, ενώ δεν έχει προλάβει το Homekit να αλλάξει τα στοιχεία του σπιτιού που μας έχει δώσει με τα σωστά που του έχουμε ζητήσει. Αυτό ονομάζεται διαδικασία thread_safe. Τέλος παράγει ειδοποιήσεις μετά την αλλαγή του επιλεγμένου “Σπιτιού” προκειμένου να μπορούμε να γνωρίζουμε αν έχουμε την δυνατότητα να προχωρήσουμε σε αλλαγές στα εξαρτήματα του “Σπιτιού” που έχουμε επιλέξει.

6.1.2 Accessories Folder

- AccessoryUpdateController: Αυτή η κλάση είναι υπεύθυνη για την ενημέρωση του Homekit framework κάθε φορά που ο χρήστης αλλάζει την τιμή ενός χαρακτηριστικού ενός αξεσουάρ μέσω της εφαρμογής. Για την επίτευξη αυτού του σκοπού κάνει τον εαυτό της χειριστή όλων των event που παράγει ένα κελί πίνακα.

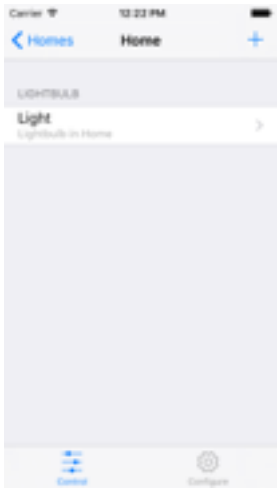
- AccessoryBrowserViewController: Αυτή η κλάση είναι υπεύθυνη για την εύρεση νέων εξαρτημάτων που υποστηρίζουν το πρωτόκολλο Homekit. Στην συγκεκριμένη εργασία χρησιμοποιείται για την προσθήκη της γέφυρας Homekit που δημιουργούμε μέσα από το NodeJS. Αυτό επιτυγχάνεται με χρήση των βιβλιοθηκών του Homekit framework.



Εικόνα 6.4 : Εύρεση αξεσουάρ

- ControlsViewController: Αυτή η κλάση είναι υπεύθυνη για την εμφάνιση των διαθέσιμων υπηρεσιών που έχουν εγγραφεί στο επιλεγμένο “Σπίτι”, οι οποίες αντιστοιχίζονται σε εξαρτήματα. Οι υπηρεσίες αυτές προσδιορίζουν τα χαρακτηριστικά που θα έχει το κάθε εξάρτημα. Με την επιλογή οποιασδήποτε υπηρεσίας ανοίγει μια νέα σελίδα η οποία περιέχει όλα τα χειριστήρια για τα διαθέσιμα χαρακτηριστικά κάθε υπηρεσίας.

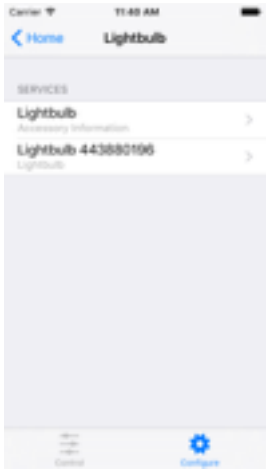
- ControlsTableViewDataSource: Αυτή η κλάση είναι υπεύθυνη για την εύρεση και την τοποθέτηση στον πίνακα που εμφανίζεται στον οθόνη ControlsView των διαθέσιμων υπηρεσιών που προσφέρει το κάθε αξεσουάρ. Επίσης έχει την δυνατότητα να παρουσιάζει με αχνό χρώμα τις



Εικόνα 6.5 : Διαθέσιμες υπηρεσίες

υπηρεσίες που δεν βρίσκει διαθέσιμες και σε εμβέλεια, προκειμένου να ειδοποιεί τον χρήστη ότι δεν μπορεί να πραγματοποιήσει κάποια ενέργεια στις υπηρεσίες των εξαρτημάτων αυτών.

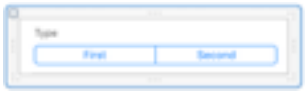
- Services folder: Αυτός ο φάκελος περιέχει κλάσεις οι οποίες είναι υπεύθυνες για τον χειρισμό των κελιών που τοποθετούνται στον πίνακα ControlsView, κλάσεις που είναι υπεύθυνες για την εμφάνιση των διαθέσιμων υπηρεσιών σε κάθε αξεσουάρ στην καρτέλα Configure και κλάσεις υπεύθυνες για κάθε είδος κελιού το οποίο περιέχει μηχανισμούς χειρισμού των διαφόρων χαρακτηριστικών ενός εξαρτήματος. Παρακάτω θα βρείτε την ανάλυση των κλάσεων αυτών.
  - ServiceCell: Αυτή η κλάση είναι υπεύθυνη για τον χειρισμό των κελιών που εμφανίζονται στον πίνακα ControlsView και είναι υπεύθυνη για την εμφάνιση των πληροφοριών μιας υπηρεσίας μέσα στα κατάλληλα πεδία ενός κελιού.
  - ServicesViewController: Αυτή η κλάση είναι υπεύθυνη για την δημιουργία του γραφικού περιβάλλοντος το οποίο εμφανίζεται στην καρτέλα Configure μετά την επιλογή κάποιου εξαρτήματος και είναι υπεύθυνη για την εμφάνιση των υπηρεσιών που υποστηρίζει το κάθε εξάρτημα. Αυτό το επιτυγχάνει καθώς είναι δηλωμένη ως ελεγκτής πινάκων με αποτέλεσμα εκτός από δημιουργία γραφικού περιβάλλοντος να μπορεί να παρέχει και την δυνατότητα για να δημιουργεί κελιά σε έναν πίνακα.



Εικόνα 6.6 : Διαθέσιμες υπηρεσίες στο tab configure

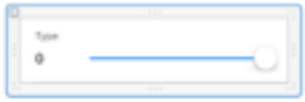
- Characteristic Cells folder: Αυτός ο φάκελος περιέχει τις κλάσεις που χειρίζονται την τοποθέτηση των κατάλληλων λεκτικών στα κελιά των πινάκων που χρησιμοποιούνται για τον χειρισμό των λειτουργιών των εξαρτημάτων ενώ είναι υπεύθυνες και για την ενημέρωση της κλάσης CharacteristicCell, της οποίας είναι “παιδιά”, για την οποιαδήποτε αλλαγή πραγματοποιείται στα χειριστήρια των λειτουργιών των εξαρτημάτων.

- CharacteristicCell: Αυτή η κλάση χρησιμοποιείται για την εμφάνιση των πληροφοριών που αντιστοιχούν σε ένα χαρακτηριστικό μέσα σε ένα κελί πίνακα. Διαθέτει μεθόδους ενημέρωσης του Homekit για την αλλαγή των τιμών ενός χαρακτηριστικού. Δημιουργούμε τα παρακάτω παιδιά της κλάσης αυτής προκειμένου να εμφανίσουμε πολλαπλά χειριστήρια ελέγχου για τα διαφορετικά χαρακτηριστικά.
- SegmentedCharacteristicCell: Αυτό το χειριστήριο χρησιμοποιείται για τον χειρισμό λειτουργιών ενός εξαρτήματος οι οποίες αποτελούνται από προαποφασισμένες εντολές προς το εξάρτημα. Αυτή η κλάση είναι υπεύθυνη για την ειδοποίηση της κλάσης CharacteristicCell της οποίας είναι παιδί, προκειμένου να γίνει ενημέρωση του Homekit για την αλλαγή που ζήτησε ο χρήστης.



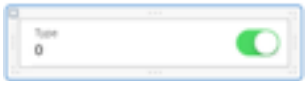
Εικόνα 6.7 : Segmented Cell

- SliderCharacteristicCell: Αυτό το χειριστήριο χρησιμοποιείται για τον έλεγχο λειτουργιών εξαρτημάτων οι οποίες προσδιορίζονται με αριθμητικές τιμές. Όπως για παράδειγμα την ταχύτητα ενός ανεμιστήρα ή την θερμοκρασία ενεργοποίησης ενός θερμοστάτη. Αυτή η κλάση είναι υπεύθυνη για την ειδοποίηση της κλάσης CharacteristicCell της οποίας είναι παιδί, προκειμένου να γίνει ενημέρωση του Homekit για την αλλαγή που ζήτησε ο χρήστης.



Εικόνα 6.8 : Slider Cell

- SwitchCharacteristicCell: Αυτό το χειριστήριο χρησιμοποιείται για τον έλεγχο λειτουργιών εξαρτημάτων τα οποία μπορούν να προσδιορισθούν με λογικές καταστάσεις 0 ή 1. Όπως για παράδειγμα την κατάσταση ενός διακόπτη. Αυτή η κλάση είναι υπεύθυνη για την ειδοποίηση της κλάσης CharacteristicCell της οποίας είναι παιδί, προκειμένου να γίνει ενημέρωση του Homekit για την αλλαγή που ζήτησε ο χρήστης.



Εικόνα 6.9 : Switch Cell

- ColorPickerCharacteristicCell: Αυτό το χειριστήριο χρησιμοποιείται για τον έλεγχο λειτουργιών εξαρτημάτων λάμπας τα οποία περιέχουν ιδιότητες αλλαγής χρώματος φωτισμού. Χρησιμοποιεί ένα “ψηφιδωτό” χρωμάτων προκειμένου να δίνει την δυνατότητα στον χρήστη να επιλέγει το χρώμα που θέλει να επιλέξει για την λάμπα, ενώ μέσω του ίδιου χειριστηρίου του δείχνει το τρέχον χρώμα. Αυτή η κλάση είναι υπεύθυνη για την ειδοποίηση της κλάσης CharacteristicCell της οποίας είναι παιδί, προκειμένου να γίνει ενημέρωση του Homekit για την αλλαγή που ζήτησε ο χρήστης. Η κλάση αυτή παραβαίνει τους κανόνες των υπολοίπων κλάσεων καθώς δεν διαχειρίζεται μόνο ένα χαρακτηριστικό, αλλά τα τρία χαρακτηριστικά που είναι απαραίτητα για τον έλεγχο των χρωμάτων σε μια λάμπα. Αυτά είναι το Hue(απόχρωση), Saturation(κορεσμός) και το Brightness(φωτεινότητα). Ο επιλογέας συρσίματος ελέγχει την φωτεινότητα της παλέτας των χρωμάτων.

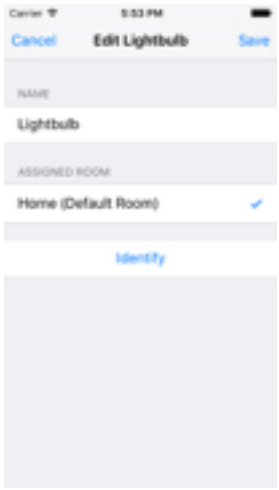
Εικόνα 6.10 : Color pick Cell

- CharacteristicViewController: Αυτή η κλάση είναι υπεύθυνη για το γραφικό περιβάλλον της οθόνης που περιέχει τον πίνακα με τα χειριστήρια των λειτουργιών που υποστηρίζει ένα αξεσουάρ. Είναι υπεύθυνο για την δημιουργία του πίνακα και την τοποθέτηση του στην οθόνη αλλά όχι για την δημιουργία των κελιών που υπάρχουν στον πίνακα.
- CharacteristicTableViewDataSource: Αυτή η κλάση είναι υπεύθυνη για την δημιουργία και την τοποθέτηση των κελιών στον πίνακα που διαχειρίζεται η κλάση CharacteristicsViewController. Επικοινωνεί με το Homekit από το οποίο παίρνει τις διαθέσιμες λειτουργίες του κάθε εξαρτήματος και εμφανίζει τα κατάλληλα χειριστήρια προκειμένου να επιτρέψει τον έλεγχο του. Ενώ δέχεται από την προηγούμενη οθόνη, την ControlsViewController το επιλεγμένο

Εικόνα 6.11 : Χαρακτηριστικά ενός αξεσουάρ στο tab control

αξεσουάρ από τον χρήστη, προκειμένου να εμφανίσει τα κατάλληλα χειριστήρια. Αυτό το επιτυγχάνει κατά την διαδικασία μεταβίβασης σε αυτή την σελίδα με την μεταβλητή service, την οποία έχει δηλώσει ως δημόσια σε κλάσεις που την χρησιμοποιούν και η οποία του περνιέται από την κλάση CharacteristicsViewController, η οποία έχει την ίδια μεταβλητή δηλωμένη και αυτή με την σειρά της.

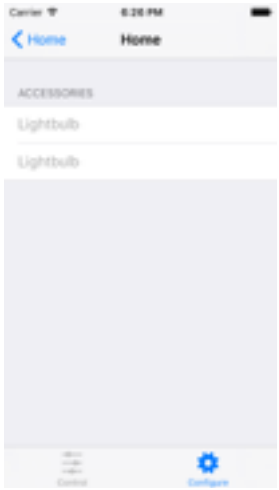
- ModifyAccessoryViewController: Αυτή η κλάση είναι υπεύθυνη για τον χειρισμό των δεδομένων των κελιών και του πίνακα που εμφανίζεται στην οθόνη που χρησιμοποιούμε για την μετονομασία ενός εξαρτήματος ή την ανάθεση του σε διαφορετικά δωμάτια από το προκαθορισμένο. Αποτελεί δε χειριστή αντικειμένων πίνακα ενώ επικοινωνεί και με το Homekit για την επεξεργασία του κάθε εξαρτήματος. Το εξάρτημα που επεξεργάζεται κάθε φορά ενημερώνεται με την χρήση μιας δημόσιας μεταβλητής, σε κλάσεις που την χρησιμοποιούν με το όνομα accessory.



Εικόνα 6.12 : Επεξεργασία ονόματος και δωματίου αξεσουάρ

6.1.3 Rooms folder

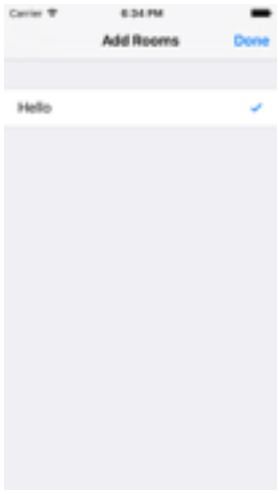
- RoomViewController: Αυτή η κλάση χρησιμοποιείται για τον έλεγχο και την δημιουργία της οθόνης ελέγχου των δημιουργημένων “δωματίων”. Από αυτήν την οθόνη μπορούμε να επιλέξουμε από τα ενεργά εξαρτήματα και να τα αναθέσουμε στο επιλεγμένο “δωμάτιο”, προκειμένου να οργανώσουμε το εικονικό “Σπίτι” μας. Έτσι γίνεται πιο εύκολος ο έλεγχος του “Σπιτιού” μέσω φωνητικών εντολών μιας και μπορούμε να ανοίξουμε με μια φράση όλες τις συσκευές ενός “δωματίου”.



Εικόνα 6.13 : Παράθυρο επιλεγμένου δωματίου

6.1.4 Zones folder

- AddRoomViewController: Αυτή η κλάση ελέγχει ένα παράθυρο το οποίο εμφανίζεται μέσα από το μενού επεξεργασίας μιας ζώνης προκειμένου να εμφανίσει τα διαθέσιμα δωμάτια, τα οποία έχουν εισαχθεί στο σπίτι για να ομαδοποιηθούν σε μια ζώνη. Επίσης είναι εγγεγραμμένη ως ένας ελεγκτής πινάκων προκειμένου να ελέγχει τον πίνακα που περιλαμβάνει τα διαθέσιμα δωμάτια. Ενώ ξέρει σε ποια ζώνη θα προστεθούν τα δωμάτια με μια δημόσια - σε όσους χρησιμοποιούν την κλάση της - μεταβλητή η οποία περιέχει την ζώνη που θα προστεθούν τα δωμάτια κάθε φορά.



Εικόνα 6.14 : Παράθυρο ζωνών

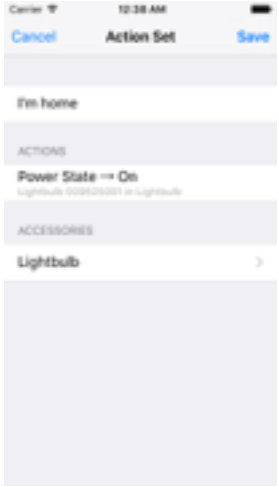
- ZoneViewController: Αυτή η κλάση χρησιμοποιείται για τον έλεγχο και την δημιουργία της οθόνης ελέγχου των δημιουργημένων “ζωνών”. Από αυτήν την οθόνη μπορούμε να δούμε τα δωμάτια που περιέχονται στην κάθε ζώνη και να αφαιρέσουμε όποιο είναι περιττό, αλλά και να καλέσουμε την οθόνη AddRoom προκειμένου να προσθέσουμε και άλλα.

Εικόνα 6.15 : Παράθυρο επιλεγμένης ζώνης

6.1.5 Action Sets folder

- ActionSetCreator: Αυτή η κλάση παρέχει μεθόδους οι οποίες διευκολύνουν την δημιουργία group πολλαπλών ενεργειών και την αποθήκευση τους σε σπίτια. Χρησιμοποιούμε αυτή την κλάση προκειμένου να κρατάμε ξεχωριστές τις ενέργειες που δημιουργούν τα action sets, οι οποίες εκτελούνται σε πολλαπλά thread από την εκτέλεση κώδικα UI ο οποίος εκτελείται σε ένα thread. Επίσης παρέχει υπηρεσίες όπως η εξέταση της τιμής ενός

χαρακτηριστικού, μετά την ενεργοποίηση ενός action set σε ένα αξεσουάρ. Ακόμα δημιουργεί action set και επιτρέπει την επιτόπια ανάθεση ενεργειών στο action set ή την τοποθέτηση ενεργειών σε μεταγενέστερο χρόνο. Αυτή η κλάση επίσης αποτελεί διαμεσολαβητή κελιού πίνακα προκειμένου να αποκαλύπτει έτσι όλες τις επιπλέον ενέργειες της στις κλάσεις που ελέγχουν κελιά πίνακα με έναν πολύ απλό και λογικό τρόπο καθώς κάθε κλάση που ελέγχει κελιά πινάκων πρέπει να ορίζει έναν διαμεσολαβητή κελιού πίνακα, προκειμένου να επικοινωνεί μέσω αυτού με το thread του UI.



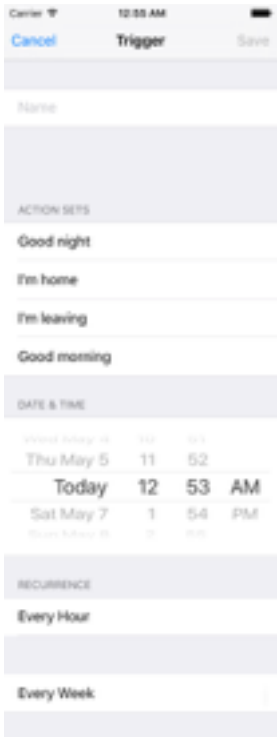
Εικόνα 6.16 : Δημιουργία action set

- ActionSetViewController: Αυτή η κλάση είναι υπεύθυνη για την δημιουργία του UI που επιτρέπει την δημιουργία νέων action set. Είναι δηλωμένη ως χειριστής πινάκων και εμφανίζει όλα τα διαθέσιμα αξεσουάρ για το επιλεγμένο σπίτι, ενώ καταγράφει κάθε αλλαγή τιμής στα χαρακτηριστικά ενός εξαρτήματος και την αναθέτει στο επιλεγμένο action set. Τέλος εμφανίζει όλες τις αλλαγές που έχουν καταγραφεί στο συγκεκριμένο action set και τις εμφανίζει στην αρχική οθόνη. Ο τρόπος που επιτρέπει την αποθήκευση των αλλαγών στις τιμές του αξεσουάρ μέσα σε ένα action set είναι η ανάθεση του δικού της actionsetcreator αντικειμένου, ως τον διεκπεραιωτή στον πίνακα που εμπεριέχεται στην οθόνη τύπου ServicesViewController, που εμφανίζεται όταν επιλέξουμε οποιοδήποτε αξεσουάρ. Επιπλέον δίνει την δυνατότητα μετονομασίας του επιλεγμένου action set αλλά και αφαίρεση οποιασδήποτε ενέργειας υπάρχει μέσα στο action set.
- ActionCell: Αυτή η κλάση είναι υπεύθυνη για την δημιουργία των κελιών που περιγράφουν τις ενέργειες που περιέχονται σε ένα action set.

Εικόνα 6.17 : Κατάσταση μιας υπηρεσίας

6.1.6 Triggers Folder

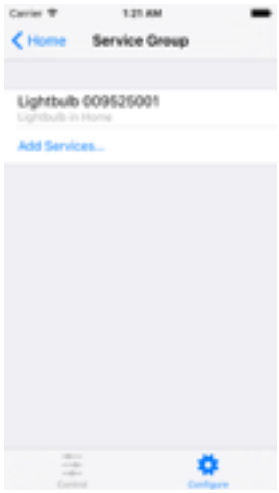
TriggerViewController: Αυτή η κλάση είναι υπεύθυνη για τον έλεγχο και την δημιουργία της οθόνης επεξεργασίας των χρονοδιακοπτών. Είναι δηλωμένη σαν ελεγκτής πινάκων και έχει την δυνατότητα να ζητάει όλα τα action set, προκειμένου να τα εμφανίζει μέσα στον πίνακα ώστε να επιλέγει ο χρήστης ποιό(ά) action set θέλει να ενεργοποιηθούν την ώρα και ημέρα που θα ορισθεί. Ενώ έχει επίσης την δυνατότητα να ενεργοποιήσει τον κάθε χρονοδιακόπτη επαναλαμβανόμενα κάθε ώρα ή κάθε εβδομάδα. Πρέπει επίσης να δώσει ένα όνομα στον χρονοδιακόπτη για να τον ταυτοποιήσει.



Εικόνα 6.18 :
Χρονοπρογραμματισμός ενεργειών

6.1.7 Service Group Folder

- ServiceGroupViewController: Αυτή η κλάση είναι υπεύθυνη για την επεξεργασία των Service Group τα οποία επιτρέπουν την δημιουργία group από εξαρτήματα ανεξαρτήτως του χώρου που βρίσκονται προκειμένου να γίνεται μαζικός έλεγχος τους ευκολότερα. Είναι δηλωμένη σαν χειριστής πινάκων και είναι υπεύθυνη για τον έλεγχο και την δημιουργία του πίνακα που εμφανίζεται στην οθόνη αυτή. Επίσης εμφανίζει, μέσα από το πάτημα του πλήκτρου Add Services, όλες τις διαθέσιμες υπηρεσίες προκειμένου να προστεθούν στο επιλεγμένο service group. Για την επίτευξη αυτού του στόχου δημιουργεί ένα παράθυρο τύπου AddServicesViewController, στο οποίο θέτει το δικό της servicegroup ως το servicegroup εκείνου του νέου



Εικόνα 6.19 :
Δημιουργία ομάδας πολλαπλών υπηρεσιών

παραθύρου, προκειμένου η κλάση AddServicesViewController να προσθέσει όλα τα επιλεγμένα εξαρτήματα στο συγκεκριμένο serviceGroup. Όταν ξανά εμφανισθεί η συγκεκριμένη οθόνη ανανεώνει το περιεχόμενο των πινάκων και βρίσκει τις υπηρεσίες των αξεσουάρ που έχουν προστεθεί. Ενώ τέλος επιτρέπει και την αφαίρεση των υπηρεσιών των αξεσουάρ από κάθε service group.

Εικόνα 6.20 : Προσθήκη υπηρεσιών σε ομάδα

- AddServicesViewController: Αυτή η κλάση είναι υπεύθυνη για την δημιουργία και τον χειρισμό της οθόνης που επιτρέπει την προσθήκη νέων υπηρεσιών σε ένα υπάρχων service group. Έχει μια δημόσια, σε όσους χρησιμοποιούν την κλάση, μεταβλητή η οποία χρησιμοποιείται για τον ορισμό του service group στο οποίο θα προστεθούν οι υπηρεσίες που θα επιλέξει ο χρήστης. Είναι δηλωμένη σαν χειριστής πινάκων και είναι υπεύθυνη για την δημιουργία και τον έλεγχο του πίνακα που εμφανίζει τις υπηρεσίες των διαφόρων εξαρτημάτων. Ενώ φροντίζει να εκτελέσει ασύγχρονα κάθε προσθήκη νέας υπηρεσίας εξαρτήματος στο service group και να καταστρέψει τον εαυτό του μόνο όταν το επιτύχει.

6.1.8 Utilities Folder

- HMCharacteristic+Properties: Αυτή η κλάση επεκτείνει τις δυνατότητες της κλάσης HMCharacteristic με μερικές μεθόδους οι οποίες μας διευκολύνουν και τις χρησιμοποιούμε διεξοδικά μέσα στην εφαρμογή μας. Όπως η αναγνώριση εάν ένα χαρακτηριστικό είναι αριθμητικό ή δυαδικό, που χρησιμοποιούμε για τον προσδιορισμό του μηχανισμού ελέγχου του χαρακτηριστικού(συρόμενος επιλογέας ή διακόπτης).

- HMCharacteristic+Readability: Αυτή η κλάση επεκτείνει τις δυνατότητες της κλάσης HMCharacteristic με μερικές μεθόδους οι οποίες μας διευκολύνουν και τις χρησιμοποιούμε διεξοδικά μέσα στην εφαρμογή μας. Όπως η εύρεση περιγραφών για κάθε είδος χαρακτηριστικού που χρησιμοποιούμε μέσα στην εφαρμογή.

- HMService+Readability: Αυτή η κλάση επεκτείνει τις δυνατότητες της κλάσης HMService με μερικές μεθόδους οι οποίες μας διευκολύνουν και τις χρησιμοποιούμε διεξοδικά μέσα στην εφαρμογή μας. Όπως η εύρεση περιγραφών για κάθε είδος υπηρεσίας που χρησιμοποιούμε μέσα στην εφαρμογή.

- NSError+Homekit: Αυτή η κλάση επεκτείνει τις δυνατότητες της κλάσης NSError με μερικές μεθόδους οι οποίες μας διευκολύνουν και τις χρησιμοποιούμε διεξοδικά μέσα στην εφαρμογή μας. Όπως η εύρεση περιγραφών για κάθε είδος μηνύματος λάθους που μπορεί να παράξει κάποια ενέργεια του χρήστη.

- NSIndexPath+ArrayIndex: Αυτή η κλάση επεκτείνει τις δυνατότητες της κλάσης NSIndexPath με μερικές μεθόδους οι οποίες μας διευκολύνουν και τις χρησιμοποιούμε διεξοδικά μέσα στην εφαρμογή μας. Συγκεκριμένα εισάγει μια μέθοδο για την εύρεση της διαδρομής του δείκτη που δείχνει στο ευρετήριο ενός αντικειμένου για ένα αντικείμενο μέσα σε πίνακα.

- UIAlertViewController+Convenience: Αυτή η κλάση επεκτείνει τις δυνατότητες της κλάσης UIAlertViewController με μερικές μεθόδους οι οποίες μας διευκολύνουν και τις χρησιμοποιούμε διεξοδικά μέσα στην εφαρμογή μας. Την χρησιμοποιούμε για την δημιουργία και επιστροφή νέων pop-up παραθύρων που χρησιμοποιούμε μέσα στην εφαρμογή για την ειδοποίηση ή για την εισαγωγή στοιχείων από και προς τον χρήστη.

- UIViewController+Convenience: Αυτή η κλάση επεκτείνει τις δυνατότητες της κλάσης UIViewController με μερικές μεθόδους οι οποίες μας διευκολύνουν και τις χρησιμοποιούμε διεξοδικά μέσα στην εφαρμογή μας. Την χρησιμοποιούμε για την εμφάνιση διαλόγων λάθους ή πληροφόρησης ή και εισαγωγής στοιχείων προς τον χρήστη για μεγαλύτερη ευκολία προκειμένου να μην επαναλαμβάνουμε συνεχώς τις μεθόδους αυτές σε κάθε κλάση και για να κρατάμε καθαρό τον κώδικα μας.

- UITableView+EmptyMessage: Αυτή η κλάση επεκτείνει τις δυνατότητες της κλάσης UITableView με μερικές μεθόδους οι οποίες μας διευκολύνουν και τις

χρησιμοποιούμε διεξοδικά μέσα στην εφαρμογή μας. Αυτή η κλάση ελέγχει έναν πίνακα και εάν βρει πως το πλήθος των στοιχείων του είναι μηδενικό προσθέτει ένα λεκτικό που αναφέρει πως δεν υπάρχουν στοιχεία για εμφάνιση.

- UITableView+Updating: Αυτή η κλάση επεκτείνει τις δυνατότητες της κλάσης UITableView με μερικές μεθόδους οι οποίες μας διευκολύνουν και τις χρησιμοποιούμε διεξοδικά μέσα στην εφαρμογή μας. Προσθέτει μια μέθοδο η οποία δέχεται μια συνάρτηση την οποία εκτελεί η μέθοδος αυτής της κλάσης όταν τα περιεχόμενα ενός πίνακα θα αλλάξουν προτού όμως αυτά αλλάξουν.

- HMHome+Properties: Αυτή η κλάση επεκτείνει τις δυνατότητες της κλάσης HMHome με μερικές μεθόδους οι οποίες μας διευκολύνουν και τις χρησιμοποιούμε διεξοδικά μέσα στην εφαρμογή μας. Για παράδειγμα προσφέρει εύκολες μεθόδους για την ομαδοποίηση όλων των δωματίων σε ένα σπίτι μέσα σε ένα NSArray όπως επίσης την δυνατότητα για αναζήτηση ενός αξεσουάρ μέσα σε ένα σπίτι.

Βιβλιογραφία

[1] ATMEL, ATmega164P/324P/644P Summary, in: ATMEL (Ed.) ATmega164P/324P/644P Summary, ATMEL, http://atmel.com, 2015.

[2] Olimex, HC-06 Product Data Sheet, in: L. Guangzhou HC Information Technology Co. (Ed.), Olimex, Website, 2011.

[3] Wikipedia, Relay, in.

[4] Wikipedia, Buck converter, in.

[5] Apple, Homekit Framework User Inteface Guidelines, in, Apple Inc, https://developer.apple.com/homekit/ui-guidelines/, 2016, pp. With HomeKit, users can utilize home automation apps on their iOS devices to control and configure the connected accessories in their homes, regardless of the manufacturer.

[6] M. Nowotny, aJSON, in: I. Matter (Ed.) aJSON, Interactive Matter, http://github.com/interactive-matter/aJson, 2010.

[7] Unknown, Introducing JSON, in, http://www.json.org/, http://www.json.org/, 2016, pp. JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 263rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

[8] Wikipedia, Bluetooth, in, Wikipedia, 2010.

[9] KhaosT, HAP-NodeJS, in, KhaosT, https://github.com/KhaosT/HAP-NodeJS/, 2016, pp. HAP-NodeJS is a Node.js implementation of HomeKit Accessory Server.

[10] Relay, in.